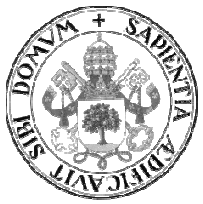


Universidad de Valladolid

E. U. de Informática (SEGOVIA)

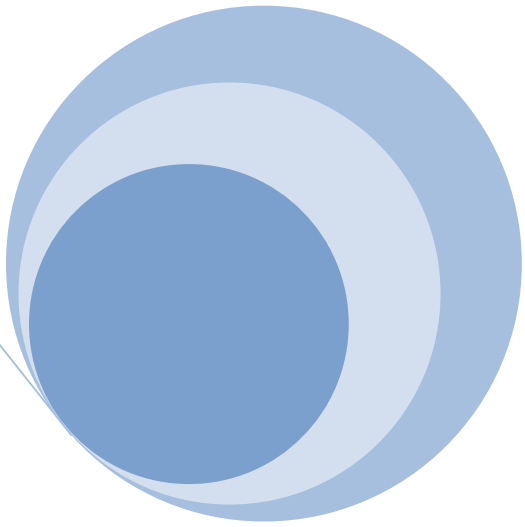
Grado en Ingeniería Informática de Servicios y Aplicaciones

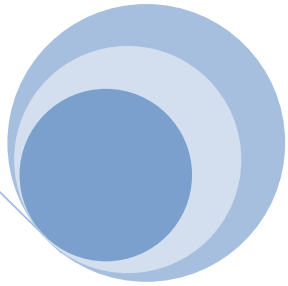
Guía Comparativa de Metodologías Ágiles

Alumno: María José Pérez Pérez

Tutor: Francisco J. González Cabrera

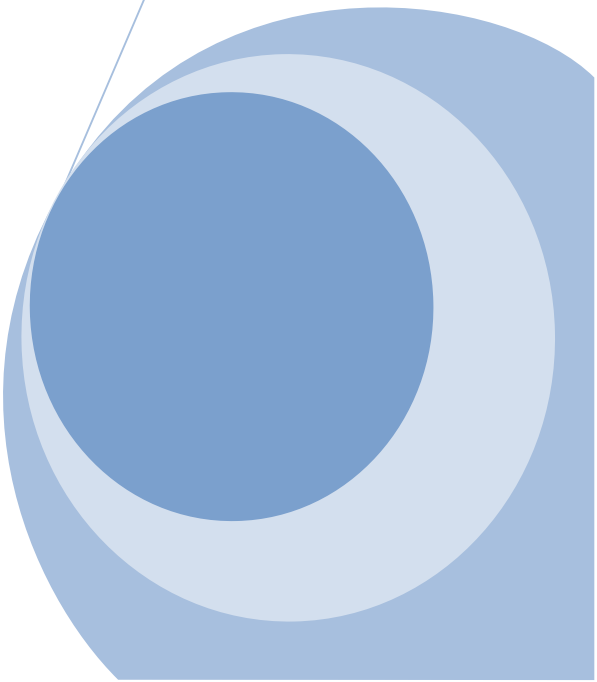
Tabla de contenidos





Capítulo 1: Introducción

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones



Contenido

Introducción

Motivación

La idea surge por el interés personal en la ingeniería del software, en concreto, por el desarrollo de software aplicando metodologías ágiles, ya que he visto que mejoran la calidad del trabajo realizado en muchos aspectos, ayudan a ofrecer un buen producto, además de tener contento al cliente y por supuesto, que el equipo trabaje cómodo. Después de comenzar a trabajar con SCRUM en mi empresa, hubo un periodo de documentación para conocer esta metodología y a la vez conocer, con menos profundidad, otras metodologías ágiles. A partir de este momento he visto la necesidad de elaborar una clasificación / comparativa que me diese unos criterios para saber qué metodología ágil se adapta mejor a un contexto de trabajo.

Las metodologías de desarrollo de software son decisivas en el éxito o fracaso de un proyecto. En general las metodologías ponen en práctica una serie de procesos comunes, que son buenas prácticas para lograr los objetivos de negocio, costes, funcionalidad, sencillez, etc. La elección de una metodología inadecuada o su mala aplicación pueden conducir a que el proyecto no llegue a su fin.

Hasta hace muy poco, se venían utilizando las llamadas metodologías tradicionales, donde los procesos son prácticamente secuenciales, están cargados de documentación lo que los hace poco flexibles frente al cambio.

Hoy en día, con un escenario en el que los requisitos cambian habitualmente es donde surge la necesidad de conocimiento sobre las metodologías ágiles, más ligeras y versátiles. En este proyecto hay un objetivo de divulgación de las metodologías ágiles más importantes, así como saber en qué contexto encajan y las reglas básicas del juego.

Varios criterios han hecho elegir esta temática como eje para el Trabajo Fin de Grado. Entre ellos el contacto reciente en mi trabajo, su importancia y las futuras aplicaciones en la vida laboral.

Objetivos del proyecto

- Seleccionar la metodologías ágiles que van a formar parte del estudio. Entender y documentar las características fundamentales de cada una de ellas.
- Seleccionar los criterios de comparación.
- Elaborar la comparativa de las metodologías ágiles anteriormente estudiadas.
- Presentación de un caso práctico utilizando SCRUM.

Es muy importante escoger una buena metodología para tener éxito en el proyecto, lo que implica muchas más cosas que sacar un producto tiempo, supone que el cliente esté satisfecho y que el equipo trabaje cómodo.

Actualmente, las metodologías ágiles están en auge dentro del desarrollo software. Dentro de la diversidad de metodologías ágiles, se trata de elaborar una comparativa que sirva para tener unos criterios y escoger una metodología ágil u otra en función del marco en el que se vaya a implantar.

Capítulo 2: Metodologías ágiles y características

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Contenido

Metodologías ágiles y características

Manifiesto Ágil

La Alianza Ágil elaboró un conjunto de doce principios comunes a las metodologías ágiles de desarrollo que se enuncian a continuación:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.

4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

7. El software que funciona es la principal medida del progreso.

8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica enaltece la agilidad.

10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.

12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

La utilización de todas las buenas prácticas enumeradas en el manifiesto ágil no implica ser ágil, sin embargo, el hecho de incumplir una de ellas te transforma en no ágil.

A la hora de elaborar el manifiesto ágil se han tenido en cuenta los siguientes puntos, dándole más valor a la primera parte que a la segunda:

1. **Se valora a los individuos y las interacciones** sobre los procesos y las herramientas.
2. **Se valoras las aplicaciones que funcionan** sobre la documentación exhaustiva.
3. **Se valora la colaboración del cliente** sobre las negociaciones contractuales.
4. **Se valora la respuesta al cambio** sobre el seguimiento de un plan.

A continuación se van a presentar las metodologías ágiles de desarrollo más exitosas. El propósito de este apartado es responder a las siguientes preguntas:

- *¿Qué es Scrum?*
- *¿Qué es XP (eXtreme Programming)?*
- *¿Kanban?*
- *¿Scrumban?*

Scrum

En Scrum un proyecto se ejecuta en bloques temporales (iteraciones-sprints) de un mes natural (pueden ser de dos o tres semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

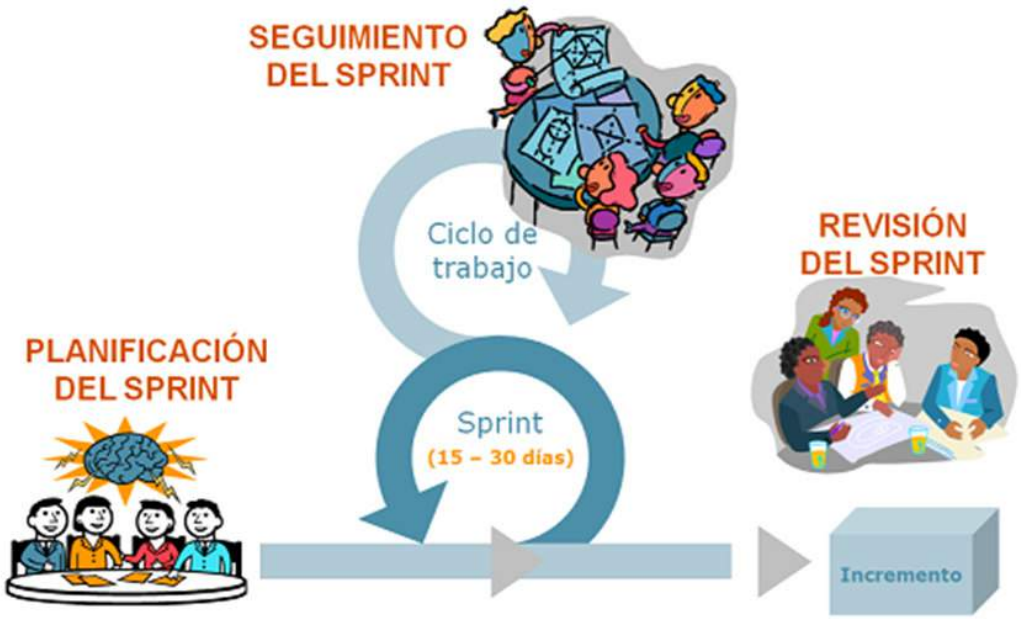


Ilustración 2-1: Ciclo de Scrum - Sprint

El Sprint es el ritmo de los ciclos de Scrum. Está delimitado por la reunión de planificación del sprint y la reunión retrospectiva. Una vez que se fija la duración del sprint es inamovible. La mayoría de los equipos eligen dos, tres o cuatro semanas de duración. Diariamente durante el sprint, el equipo realiza una reunión de seguimiento muy breve. Al final del sprint se entrega el producto al cliente en el que se incluye un incremento de la funcionalidad que tenia al inicio del sprint.

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ha priorizado los requisitos balanceando el valor que le aportan respecto a su coste y han sido divididos en iteraciones y entregas.

Actividades de la metodología Scrum

Las actividades que se llevan se plantea realizar en la metodología Scrum son las siguientes:

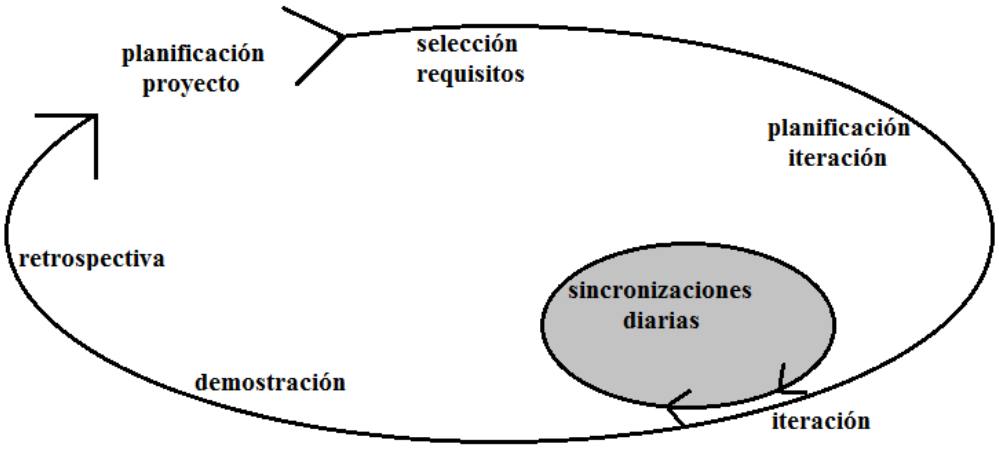


Ilustración 2-2: Actividades del proceso de Scrum

Planificación de la iteración

La planificación de las tareas a realizar en la iteración se divide en dos partes:

Primera parte de la reunión. Se realiza en un tiempo máximo 4 horas:

- El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto, pone nombre a la meta de la iteración (de manera que ayude a tomar decisiones durante su ejecución) y propone los requisitos más prioritarios a desarrollar en ella.
- El equipo examina la lista, pregunta al cliente las dudas que le surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Segunda parte de la reunión. Se realiza en un tiempo máximo 4 horas. El equipo planifica la iteración, dado que ha adquirido un compromiso, es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.

- Define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.
- Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.
- Cada miembro del equipo se asigna a las tareas que puede realizar.

Beneficios

Potenciación responsable de organizar el trabajo por parte del equipo, que es quien mejor conoce como realizarlo.

- Define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.
- Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.

Potenciación del compromiso de cada miembro con el equipo:

- Es el equipo quien asume la responsabilidad de completar en la iteración los requisitos que selecciona.
- Es cada una de las personas la que se responsabiliza de realizar las tareas a las que se asigna.

Una estimación conjunta es más fiable, dado que tiene en cuenta los diferentes conocimientos, experiencia y habilidades de los integrantes del equipo.

Ejecución de la iteración (*sprint*)

- En Scrum un proyecto se ejecuta en iteraciones de un mes natural (pueden ser de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.
- Cada día el equipo realiza una reunión de sincronización, donde cada miembro inspecciona el trabajo de los otros para poder hacer las adaptaciones necesarias, así cómo comunicar cuales son los impedimentos con que se encuentra.
- El Facilitador (*Scrum Master*) se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad. Elimina los obstáculos que el equipo no puede resolver por sí mismo. Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Recomendaciones

Para poder completar el máximo de requisitos en la iteración, se debe minimizar el número de requisitos en que el equipo trabaja simultáneamente completando primero los que den más valor al cliente. Esta forma de trabajar, que se ve facilitada por la propia estructura de la lista de tareas de la iteración, permite tener más capacidad de reacción frente a cambios o situaciones inesperadas.

Restricciones

- No se puede cambiar los requisitos de la iteración en curso.

- El hecho de no poder cambiar los requisitos de la iteración una vez iniciada facilita que el cliente cumpla con su responsabilidad de conocer qué es lo más prioritario a desarrollar, antes de iniciar la iteración.

Terminación anormal de la iteración

Sólo en situaciones muy excepcionales el cliente o el equipo pueden solicitar una terminación anormal de la iteración. Esto puede suceder si, por ejemplo, el contexto del proyecto ha cambiado enormemente y no es posible esperar al final de la iteración para aplicar cambios, o si el equipo encuentra que es imposible cumplir con el compromiso adquirido. En ese caso, se dará por finalizada la iteración y se dará inicio a otra mediante una reunión de planificación de la iteración.

Reunión diaria de sincronización del equipo (*Scrum daily meeting*)

El objetivo de esta reunión es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad.

Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para al finalizar la reunión poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso conjunto que el equipo adquirió para la iteración (en la reunión de planificación de la iteración).

Cada miembro del equipo debe responder las siguientes preguntas en un intervalo de tiempo de cómo máximo 15 minutos:

- *¿Qué he hecho desde la última reunión de sincronización? ¿Pude hacer todo lo que tenía planeado? ¿Cuál fue el problema?*
- *¿Qué voy a hacer a partir de este momento?*
- *¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto?*

Como apoyo a la reunión, el equipo cuenta con la lista de tareas de la iteración, donde se actualiza el estado y el esfuerzo pendiente para cada tarea, así como con el gráfico de horas pendientes en la iteración.

Se actualiza la gráfica burndown con el trabajo realizado.

Recomendaciones

- Realizar la reunión diaria de sincronización de pie, para que los miembros del equipo no se relajen ni se extiendan en más detalles de los necesarios.
- Realizar las reuniones de colaboración entre miembros del equipo justo después de la de sincronización.

Demostración de requisitos completados (*Sprint Demonstration*)

- Reunión informal donde el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo, haciendo un recorrido por ellos lo más real y cercano posible al objetivo que se pretende cubrir.
- En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
- Se realiza en un tiempo máximo 4 horas.

Beneficios

- El cliente puede ver de manera objetiva cómo han sido desarrollados los requisitos que proporcionó, ver si se cumplen sus expectativas, entender más qué es lo que necesita y tomar mejores decisiones respecto al proyecto.
- El equipo puede ver si realmente entendió cuáles eran los requisitos que solicitó el cliente y ver en qué puntos hay que mejorar la comunicación entre ambos.
- El equipo se siente más satisfecho cuando puede ir mostrando los resultados que va obteniendo. No está meses trabajando sin poder exhibir su obra.

Retrospectiva (*Sprint Retrospective*)

El equipo analiza cómo ha sido su manera de trabajar durante la iteración, qué cosas han funcionado bien, cuáles hay que mejorar, qué cosas quiere probar hacer en la siguiente iteración, qué se ha aprendido y cuáles son los problemas que podrían impedirle progresar adecuadamente, con el objetivo de mejorar de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados que el propio equipo no pueda resolver por sí mismo.

Se realiza en un tiempo máximo 3 horas.

Beneficios

- Incrementa la productividad y el aprendizaje del equipo de manera sistemática, iteración a iteración, con resultados a corto plazo.

Replanificación del proyecto

Durante el transcurso de una iteración, el cliente va trabajando en la lista de requisitos priorizada del producto o proyecto, añadiendo requisitos, modificándolos,

eliminándolos, repriorizándolos, cambiando el contenido de iteraciones y definiendo un calendario de entregas que se ajuste mejor a sus nuevas necesidades.

Los cambios en la lista de requisitos pueden ser debidos a:

- Modificaciones que el cliente solicita tras la demostración que el equipo realiza al final de cada iteración sobre los resultados obtenidos, ahora que el cliente entiende mejor el producto o proyecto.
- Cambios en el contexto del proyecto (sacar al mercado un producto antes que su competidor, hacer frente a urgencias o nuevas peticiones de clientes, etc.).
- Nuevos requisitos o tareas como resultado de nuevos riesgos en el proyecto.

Para realizar esta tarea, el cliente colabora con el equipo y obtiene de él la estimación de costes de desarrollo para completar cada requisito. El equipo ajusta el factor de complejidad, el coste para completar los requisitos y su velocidad de desarrollo en función de la experiencia adquirida hasta ese momento en el proyecto.

Hay que notar que el equipo sigue trabajando con los requisitos de la iteración en curso, (que de hecho eran los más prioritarios al iniciar la iteración). No es posible cambiar los requisitos que se desarrollan durante la iteración. En la reunión de planificación de la iteración el cliente presentará la nueva lista de requisitos para que sea desarrollada.

Beneficios

De manera sistemática, iteración a iteración, se obtienen los siguientes beneficios:

- El cliente puede tomar decisiones con tiempo respecto al progreso del proyecto y posibles desviaciones:

    - Replanificar el proyecto para obtener un nuevo calendario de entregas que cumpla con sus necesidades actuales.
    - Incorporar nuevos recursos.
    - Cancelar el proyecto con los requisitos completados hasta el momento plenamente operativos, si el beneficio pendiente de obtener es menor que el coste de desarrollo.

- El plan de proyecto se actualiza con la velocidad de desarrollo del equipo, se evitan sorpresas de última hora.

Roles

Cuando se aplica la metodología Scrum se determinan las responsabilidades siguientes:

No hay un jefe de proyecto. Las responsabilidades del tradicional jefe de proyecto se distribuyen a los siguientes roles de un equipo Scrum:

- El cliente o Product Owner
- Scrum master o facilitador
- Resto del equipo

Cliente (*Product Owner*)

Las responsabilidades del Cliente (que puede ser interno o externo a la organización) son:

- Ser el representante de todas las personas interesadas en los resultados del proyecto (internas o externas a la organización, promotores del proyecto y usuarios finales) y actuar como interlocutor único ante el equipo, con autoridad para tomar decisiones.
- Definir los objetivos del producto o proyecto.
- Dirigir los resultados del proyecto.

    - Es el propietario de la planificación del proyecto: crea y mantiene la lista priorizada con los requisitos necesarios para cubrir los objetivos del producto o proyecto, conoce el valor que aportará cada.
    - Divide la lista de requisitos estableciendo un calendario de entregas.
    - Antes de iniciar cada iteración replanifica el proyecto en función de los requisitos que aportan más valor en ese momento, de los requisitos completados en la iteración anterior y del contexto del proyecto en ese momento (demandas del mercado, movimientos de la competencia, etc.).

- Participar en la reunión de planificación de iteración, proponiendo los requisitos más prioritarios a desarrollar, respondiendo a las dudas del equipo y detallando los requisitos que el equipo se compromete a hacer.
- Estar disponible durante el curso de la iteración para responder a las preguntas que puedan aparecer.
- No cambiar los requisitos que se están desarrollando en una iteración, una vez está iniciada.
- Participar en la reunión de demostración de la iteración, revisando los requisitos completados.

Facilitador (*Scrum Master*)

Lidera al equipo llevando a cabo las siguientes responsabilidades:

- Velar que todos los participantes del proyecto sigan las reglas y proceso de Scrum, encajándolas en la cultura de la organización, y guiar la colaboración del equipo con el cliente de manera que las sinergias sean máximas. Esto implica:

    - Asegurar que la lista de requisitos priorizada esté preparada antes de la siguiente iteración.
    - Facilitar las reuniones de Scrum (planificación de la iteración, reuniones diarias de sincronización del equipo, demostración, retrospectiva), de manera que sean productivas y consigan sus objetivos.

- Quitar los impedimentos que el equipo tiene en su camino para conseguir el objetivo de cada iteración (proporcionar un resultado útil al cliente de la manera más efectiva) y poder finalizar el proyecto con éxito. Estos obstáculos se identifican de manera sistemática en las reuniones diarias de sincronización del equipo y en las reuniones de retrospectiva.
- Proteger y aislar al equipo de interrupciones externas durante la ejecución de la iteración (introducción de nuevos requisitos, "secuestro" no previsto de un miembro del equipo, etc.). De esta manera, el equipo puede mantener su productividad y el compromiso que adquirió sobre los requisitos que completaría en la iteración..
- Asegurar que los requisitos se desarrollan con calidad.
- Enseñar al equipo a auto gestionarse. No da respuestas, si no que guía al equipo con preguntas para que descubra por sí mismo una solución.

Equipo (*Team*)

Grupo de personas que de manera conjunta desarrollan el producto del proyecto. Comparten la responsabilidad del trabajo que realizan (así como de su calidad) en cada iteración y en el proyecto. El tamaño del equipo está entre 5 y 9 personas. Por debajo de 5 personas cualquier imprevisto o interrupción sobre un miembro del equipo compromete seriamente el compromiso que han adquirido y, por tanto, el resultado que se va a entregar al cliente al finalizar la iteración. Por encima de 9 personas, la comunicación y colaboración entre todos los miembros se hace más difícil y se forman subgrupos.

Es un equipo auto gestionado, que realiza de manera conjunta las siguientes actividades:

- Seleccionar los requisitos que se compromete a completar en una iteración, de forma que estén preparados para ser entregados al cliente.
- En la lista de requisitos priorizados del producto, estimar la complejidad de cada uno de ellos.
- En la reunión de planificación de la iteración decide cómo va a realizar su trabajo:

    - Seleccionar los requisitos que pueden completar en cada iteración, realizando al cliente las preguntas necesarias.

  - Identificar todas las tareas necesarias para completar cada requisito.
  - Estimar el esfuerzo necesario para realizar cada tarea.
  - Cada miembro del equipo se asigna a las tareas.

- Durante la iteración, trabajar de manera conjunta para conseguir los objetivos de la iteración. Cada especialista lidera el trabajo en su área y el resto colaboran si es necesario para poder completar un requisito.
- Al finalizar la iteración:

  - Demostrar al cliente los requisitos completados en cada iteración.
  - Hacer una retrospectiva final de cada iteración para mejorar de forma continua su manera de trabajar.

El equipo es multidisciplinar:

- Los miembros del equipo tienen las habilidades necesarias para poder identificar y ejecutar todas las tareas que permiten proporcionar al cliente los requisitos comprometidos en la iteración.
- Tienen que depender lo mínimo de personas externas al equipo, de manera que el compromiso que adquieren en cada iteración no se ponga en peligro.
- Se crea una sinergia que permite que el resultado sea más rico al nutrirse de las diferentes experiencias, conocimientos y habilidades de todos. Colaboración creativa.

Los miembros del equipo deben dedicarse al proyecto a tiempo completo para evitar dañar su productividad por cambios de tareas en diferentes proyectos, para evitar interrupciones externas y así poder mantener el compromiso que adquieren en cada iteración.

Todos los miembros del equipo trabajan en la misma localización física, para poder maximizar la comunicación entre ellos mediante conversaciones cara a cara, diagramas en pizarras, etc. De esta manera se minimizan otros canales de comunicación menos eficientes, que hacen que las tareas se transformen en un “pasa pelota” o que hacen perder el tiempo en el establecimiento de la comunicación

El equipo debe ser estable durante el proyecto, sus miembros deben cambiar lo mínimo posible, para poder aprovechar el esfuerzo que les ha costado construir sus relaciones interpersonales, engranarse y establecer su organización del trabajo.

Herramientas

Entre las herramientas que son necesarias en la aplicación de Scrum se encuentran:

Lista de requisitos priorizada (*Product Backlog*)

La lista de requisitos priorizada representa las expectativas del cliente respecto a los objetivos y entregas del producto o proyecto. El cliente es el responsable de crear y gestionar la lista (con la ayuda del Facilitador (Scrum Master) y del equipo, quien proporciona el coste estimado de completar cada requisito). Al reflejar las expectativas del cliente, esta lista permite involucrarle en la dirección de los resultados del producto o proyecto.

- Contiene los requisitos de alto nivel del producto o proyecto. Para cada requisito se indica el valor que aporta al cliente y el coste estimado de completarlo. La lista está priorizada balanceando el valor que cada requisito aporta al negocio frente al coste estimado que tiene su desarrollo.
- En la lista se indican las posibles iteraciones y las entregas esperadas por el cliente (los puntos en los cuales desea que se le entreguen los requisitos completados hasta ese momento), en función de la velocidad de desarrollo del (los) equipo(s) que trabajará(n) en el proyecto.
- La lista también tiene que considerar los riesgos del proyecto e incluir los requisitos o tareas necesarios para mitigarlos.

Antes de iniciar la primera iteración, el cliente debe tener definida la meta del producto o proyecto y la lista de requisitos creada. No es necesario que la lista sea completa ni que todos los requisitos estén detallados al mismo nivel. Basta con que estén identificados y con suficiente detalle los requisitos más prioritarios con los que el equipo empezará a trabajar. Los requisitos de iteraciones futuras pueden ser mucho más amplios y generales. La incertidumbre y complejidad propia de un proyecto hacen conveniente no detallar todos los requisitos hasta que su desarrollo esté próximo. De esta manera, el esfuerzo de recoger, detallar y desarrollar el resto de requisitos (menos prioritarios) está repartido en el período de ejecución del proyecto. Esto produce varias ventajas:

- Se evita caer en parálisis de análisis al inicio del proyecto, de manera que se puede iniciar antes el desarrollo y el cliente puede empezar a obtener resultados útiles.
- Se evita analizar en detalle requisitos no prioritarios que podrían cambiar durante el transcurso del proyecto, dado que se conocerá mejor cuál ha de ser el resultado a conseguir, o bien porque podrían ser reemplazados por otros.
- Puede llegar a un punto del proyecto en que no valga la pena analizar ni desarrollar los requisitos restantes.

En el caso del desarrollo de un producto, la lista va evolucionando durante toda la vida del producto. En el caso de un proyecto, conforme éste avance irán apareciendo los requisitos menos prioritarios que falten.

El cliente y el equipo tienen que acordar la definición de “completado” de los requisitos, qué será lo que el equipo habrá realizado para considerar que el producto esté preparado

para ser entregado al cliente al finalizar cada iteración, de manera que no haya tareas pendientes que puedan impedir utilizar los resultados del proyecto lo antes posible. De este modo, el cliente podrá tomar decisiones correctas cuando al final de cada iteración el equipo le haga una demostración de los requisitos completados (por ejemplo, solicitar una entrega del producto).

Cuando el cliente solicita una entrega de los requisitos completados hasta ese momento, el equipo puede necesitar añadir una iteración de entrega, más corta que las iteraciones habituales, donde realizar alguna tarea que no ha sido necesaria o posible hasta el momento de la entrega final.

Lista de tareas de la iteración (*Sprint Backlog*)

Lista de tareas que el equipo elabora como plan para completar los requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de incremento de producto preparado para ser entregado.

Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto.

La lista contiene las tareas, el esfuerzo pendiente para finalizarlas y la auto-asignación que han hecho los miembros del equipo.

El progreso de la iteración y su velocidad con respecto a tareas u horas pendientes se muestra mediante un gráfico de trabajo pendiente (gráfica burndown).

Gráficos de trabajo pendiente (*Burndown*)

Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los requisitos. Permite extrapolar si el Equipo podrá completar el trabajo en el tiempo estimado.

Es una gráfica que en un simple vistazo muestra la evolución del equipo respecto a los requisitos del usuario y muestra cuando se espera terminar:

- Cuanto trabajo ha sido hecho
- Cuanto trabajo queda por hacer
- Velocidad del equipo
- Fecha fin esperada

Ilustración 2-3: Gráfica burndown

- En el eje Y cantidad de trabajo pendiente de terminar
- En el eje X tiempo (iteraciones)

Este tipo de gráfico permite realizar diversas simulaciones: ver cómo se aplazan las fechas de entrega si se le añaden requisitos, ver cómo se avanzan si se le quitan requisitos o se añade otro equipo, etc.

XP (eXtreme Programming)

Típicamente un proyecto con XP lleva 10 a 15 ciclos o iteraciones

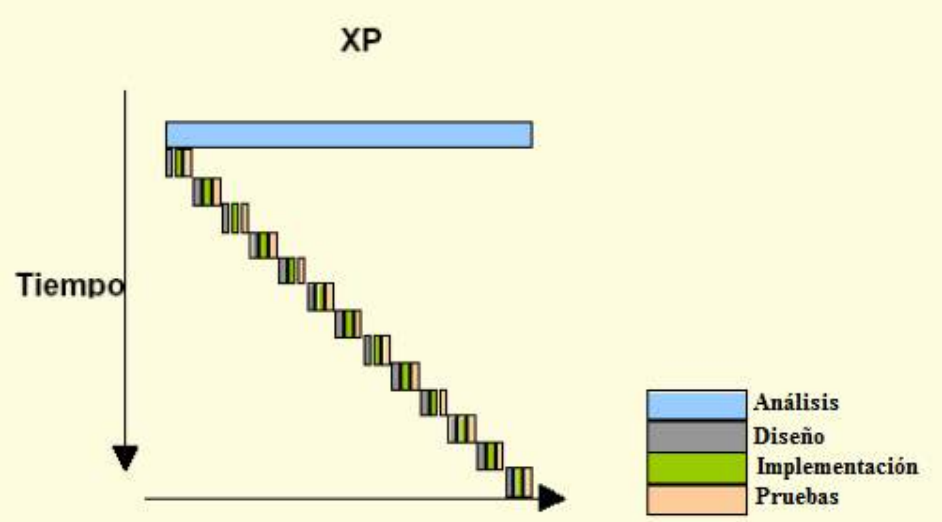


Ilustración 2-4: Ciclos XP

Fases

Fase de exploración

Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas "historias de usuario". Los programadores estimas los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que están basadas en datos de muy alto nivel), y podrían variar cuando se analicen en más detalle en cada iteración.

Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Fase de planificación

La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas que se detallará en la sección "Reglas y Practicas".

Fase de iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo.

Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

Fase de puesta en producción

Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa.

En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.

Reglas de XP

Planificación

La metodología XP plantea la planificación como un dialogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando “Historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”. Una vez obtenidas las “historias de usuarios”, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas tiene “riesgos” que no establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (“spikes”), para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas (“Release Plan”) en los que todos estén de acuerdo. Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas “historias de usuarios”.

Los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:

- Simplicidad del plan. No se espera que un plan requiera de un "gurú" con complicados sistemas de gerenciamiento de proyectos.
- Los planes son realizados por las mismas personas que realizarán el trabajo.
- Los planes no son predicciones del futuro, sino simplemente la mejor estimación de cómo saldrán las cosas. Los planes son útiles, pero necesitan ser cambiados cuando las circunstancias lo requieren. De otra manera, se termina en situaciones en las que el plan y la realidad no coinciden, y en estos casos, el plan es totalmente inútil.

Los conceptos básicos de esta planificación son los siguientes:

Historias de usuario

Las "Historias de usuarios" sustituyen a los documentos de especificación funcional, y a los "casos de uso". Estas "historias" son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

Plan de entregas (*Release Plan*)

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión "Juego de planeamiento" ("Planning game"), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, "Planning meeting" o "Planning workshop"). Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

Después de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

Reuniones diarias de seguimiento

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar. Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

Diseño

La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

Simplicidad

Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

Características fundamentales del código: Testeable, legible, comprensible y explicable.

Metáforas

Una “metáfora” es algo que todos entienden, sin necesidad de mayores explicaciones. La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Por ejemplo,

puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo.

Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta "metáfora", para que puedan dialogar en un "mismo idioma". Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Solución "spike"

Una solución "spike", es una solución muy simple para plantear posible soluciones, de manera, que solamente se aborda el problema en concreto y se aísla de otro tipo de preocupaciones.

Refactorización

La recodificación consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, más clara y eficientemente.. Las metodologías de XP sugieren recodificar cada vez que sea necesario. Si bien, puede parecer una pérdida de tiempo innecesaria en el plazo inmediato, los resultados de ésta práctica tienen sus frutos en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad. La filosofía que se persigue es, como ya se mencionó, tratar de mantener el código más simple posible que implemente la funcionalidad deseada.

Implementación

Cliente disponible

Uno de los requisitos de XP es tener al cliente disponible. No solo para ayudar al equipo de desarrollo, sino para ser parte del mismo. Todas las fases de XP requieren comunicación con el cliente.

Las historias de usuario son escritas por el cliente con la ayuda de los desarrolladores, además de establecer la prioridad de las mismas. Su presencia asegura que los desarrollos cubren toda la funcionalidad descrita.

Durante la reunión de planificación el cliente negocia las historias que se incluirán en la próxima entrega, así como su duración.

El cliente es imprescindible a la hora de realizar pruebas funcionales, en caso de error él será el encargado de decidir si el código puede pasar a producción o no.

Estándares de codificación

Todos los programadores deben escribir y documentar el código en la misma manera.

El código debe seguir los estándares de codificación. Las normas de codificación ayudan a mantener el código legible y fácil de mantener y refactorizar.

Implementación dirigida por las Pruebas Unitarias

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los tests, se realizada al final del proyecto, o al final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, en el que, lo primero que se escriben los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas.

Las pruebas a las que se refiere esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona y dirige el desarrollo.

Programación en parejas

XP promueve que todo el código sea escrito en parejas trabajando en el mismo ordenador. La programación en parejas incrementa la calidad del código sin impactar en la fecha de entrega.

En contra de lo que parece, dos personas que trabajan en un mismo equipo añadirán la misma funcionalidad que dos personas trabajando por separado, excepto que el código será de mucha mayor calidad.

Adicionalmente, la programación en pares tiene las siguientes ventajas:

- La mayoría de los errores se descubren en el momento en que se codifican, ya que el código es permanentemente revisado por dos personas.
- La cantidad de defectos encontrados en las pruebas es estadísticamente menor.
- Los diseños son mejores y el código más corto.
- El equipo resuelve problemas en forma más rápida.
- Las personas aprenden significativamente más, acerca del sistema y acerca de desarrollo de software.

- El proyecto termina con más personas que conocen los detallas de cada parte del código.
- Las personas aprenden a trabajar juntas, generando mejor dinámica de grupo y haciendo que la información fluya rápidamente.
- Las personas disfrutan más de su trabajo.

Integración secuencial

Ilustración 2-5: Integración Secuencial - XP

Todos los desarrolladores necesitan trabajar siempre con la "última versión".

Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas.

Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

Propiedad colectiva del código

La propiedad colectiva anima a todos los miembros del equipo a aportar nuevas ideas. Cualquier desarrollador puede cambiar líneas de código para añadir funcionalidad, solucionar errores, mejorar el diseño o refactorizar el código.

Cuando una persona realiza un desarrollo tiene que subir al repositorio el código más sus pruebas unitarias funcionando al 100%, de manera, que otra persona que se lo descargue puede confiar en que tiene un código que funciona y desarrollar a partir del mismo.

En la práctica, la propiedad colectiva es más fiable que poner una sola persona se encarga de vigilar segmentos específicos de código. Sobre todo porque una persona

puede dejar el proyecto en cualquier momento, de esta manera, el conocimiento está repartido.

Ritmo constante

La metodología XP indica que debe llevarse un ritmo sostenido de trabajo.

Anteriormente, ésta práctica se denominaba “Semana de 40 horas”. Sin embargo, lo importante no es si se trabajan, 35, 40 o 42 horas por semana. El concepto que se desea establecer con esta práctica es el de planificar el trabajo de manera de mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Cuando un proyecto se retrasa, trabajar tiempo extra puede ser más perjudicial que beneficioso. El trabajo extra desmotiva inmediatamente al grupo e impacta en la calidad del producto. En la medida de lo posible, se debería renegociar el plan de entregas realizando una nueva reunión de planificación con el cliente, los desarrolladores y los gerentes. Adicionalmente, agregar más desarrolladores en proyectos ya avanzados no siempre resuelve el problema.

Pruebas

Pruebas Unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (“Test-driven programming”). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.

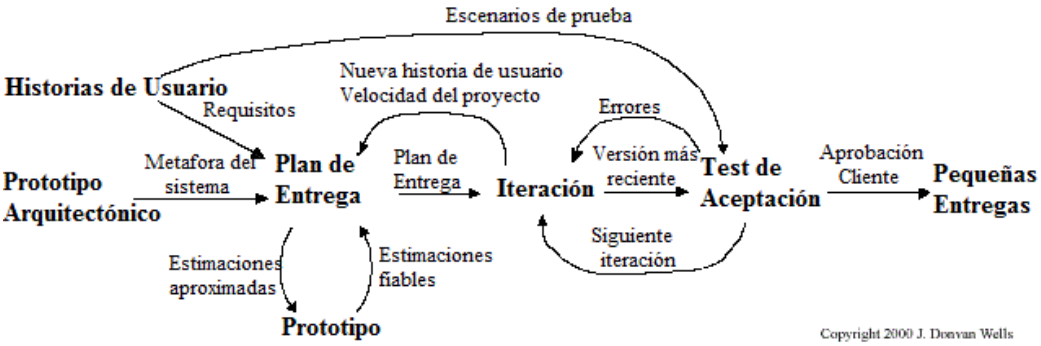


Ilustración 2-6: Proyecto XP

Valores en XP

Simplicidad: Hacer exactamente lo que se ha pedido, no más.

Comunicación: La comunicación entre los componentes del equipo XP es fundamental. Dado que la documentación es escasa, el diálogo frontal, cara a cara, entre desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.

Retroalimentación: Siempre tener en cuenta la valoración del cliente una vez que se hace una entrega e intentar mejorar haciendo cambios en el proceso si es necesario.

Coraje: Se trata que el equipo asuma la responsabilidad de su trabajo, tanto si es un éxito como un fracaso, además de ser emprendedor a la hora de implementar cambios en la aplicación (refactorizaciones…).

Roles

Cliente

El cliente es el responsable de conducir el proyecto. Define el proyecto y sus objetivos. Cuanto más preciso es su trabajo y cuanto mayor sea su involucración, mayores serán las oportunidades de éxito.

Toma decisiones de negocio y debe entender los cambios del mismo a largo del tiempo: identificando si una historia tiene el mismo valor ahora que cuando se identificó, si se puede retrasar o simplificar, además de tener en consideración las implicaciones

técnicas detectadas por los desarrolladores: es mejor cambiar el orden de la planificación de las historias de usuarios por sugerencia de los desarrolladores. Debe responder a preguntas como:

"¿Qué debería hacer esta nueva característica?"

Colaborando con los desarrolladores. Escribiendo las historias de usuario (requisitos), aclarando los detalles además de planificarlas.

"*¿Cómo sabremos cuando está lista?*"

Creando baterías de tests de aceptación del producto, verificando que las nuevas características están completas.

"*¿Cuánto podemos gastar?, ¿Cuándo comenzamos a trabajar sobre ella?*"

Participando en la reunión de planificación de las historias de usuario para la siguiente iteración.

El cliente representa el usuario final y a los intereses económicos de la empresa. Su objetivo es maximizar su inversión.

Derechos

- Maximizar la inversión, escogiendo las historias de usuario para cada iteración.
- Cambiar el alcance del proyecto para hacer frente a los cambios en la planificación, añadiendo o eliminando tareas de la iteración si las estimaciones demuestran ser incorrectas.
- Determinar que funcionalidades serán las siguientes en implementarse.
- Medir el progreso del proyecto en cualquier momento, lanzando las pruebas de aceptación.
- Detener el proyecto en cualquier momento sin perder la inversión realizada, manteniendo en producción un producto estable y continuar planificando otras funcionalidades más importantes.

Responsabilidades

- Confiar en las decisiones técnicas de los desarrolladores.
- Analizar los riesgos correctamente,
- Seleccionar las historias que aportan más valor para cada iteración.
- Definir las historias de usuario de forma precisa, permitiendo a los desarrolladores realizar estimaciones precisas.
- Participar en el equipo, dando directrices y feedback tan pronto y preciso como sea posible.

Programador

Una vez que se han comprendido las historias de usuario, el XP adjudica a los programadores la responsabilidad de tomar decisiones técnicas. Los desarrolladores estiman el tiempo que les va a tomar cada historia. Transforma las historias de usuario a código. Deben responder a preguntas como:

"*¿Cómo implementamos esta funcionalidad?*"

"*¿Cuánto nos va a llevar?*"

"*¿Cuáles son los riesgos?*"

Derechos

- Estimación de su propio trabajo, teniendo autoridad para tomar decisiones técnicas.
- Delimitar el trabajo responsabilizándose de aquellas tareas que van a ser capaces de llevar a cabo.
- Implementar la funcionalidad que cubre las necesidades del cliente.
- No tomar decisiones de negocio.

Responsabilidades

- Seguir las directrices del equipo.
- Desarrollar las funcionalidades realmente necesarias.
- Comunicación fluida con el cliente.

Encargado de Pruebas (*Tester*)

El encargado de pruebas ayuda al cliente a definir y escribir las pruebas de aceptación de las historias de usuario. Este rol en un equipo XP también es responsable realizar test periódicamente y informar de los resultados al equipo. A medida que el volumen de pruebas aumenta, el encargado de pruebas necesitará una herramienta para crear y mantener la batería de pruebas, ejecutarlas y obtener los resultados más rápidamente.

Encargado de Seguimiento (*Tracker*)

Hace el seguimiento de acuerdo a la planificación. La métrica más importante para XP es la velocidad del equipo, que se define como el tiempo ideal estimado para las tareas frente al tiempo real dedicado. Esta métrica ayuda a determinar si el proyecto está dentro del tiempo de la iteración.

Para medir la velocidad del equipo, el encargado de seguimiento pregunta uno por uno a los desarrolladores cuántas tareas ha completado. El mejor procedimiento es preguntárselo en persona, en un ambiente informal y distendido. La sinceridad es fundamental por parte de los desarrolladores, y el encargado de seguimiento no debe entrar en valoraciones.

Esta métrica ayuda a controlar el flujo del proyecto en posteriores iteraciones.

Entrenador (*Coach*)

No es un rol cubierto en todos los equipos de XP. Su papel es guiar y orientar al equipo, especialmente cuando un equipo comienza a trabajar siguiendo la metodología XP. Esto se debe a que no es fácil aplicar XP de forma consistente. Aunque son prácticas de sentido común, se necesita un tiempo para interiorizarlas. También hay situaciones especiales en las que se requiere la sabiduría de un especialista en XP para aplicar sus normas frente a un obstáculo en el proyecto.

El objetivo de un entrenador es que el equipo comprenda las directrices de XP. No se trata de que sean solamente lecciones teóricas, si no que se trata de dar ejemplo y propone ideas para mejorar.

Gestor (*Big Boss*)

Es el gerente del proyecto, debe tener una idea general del proyecto y estar familiarizado con su estado. El cliente puede asumir este papel.

Kanban

Su objetivo es gestionar de manera general como se van completando tareas, pero en los últimos años se ha utilizado en la gestión de proyectos de desarrollo software.

Las principales reglas de Kanban son las siguientes:

1. Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo.
2. Determinar el límite del “trabajo en curso” (WIP - Work In Progress).
3. Medir el tiempo en completar una tarea (Lead time).

Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo

Kanban se base en el desarrollo incremental, dividiendo el trabajo en partes. Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada tarea, y que se representa en pizarras llenas de post-it.

El trabajo se divide en partes, normalmente cada una de esas partes se escribe en un post-it y se pega en una pizarra. Los post-it suelen tener información variada, si bien, aparte de la descripción, debieran tener la estimación de la duración de la tarea.

La pizarra tiene tantas columnas como estados por los que puede pasar la tarea (ejemplo, en espera de ser desarrollada, en análisis, en diseño, etc.).

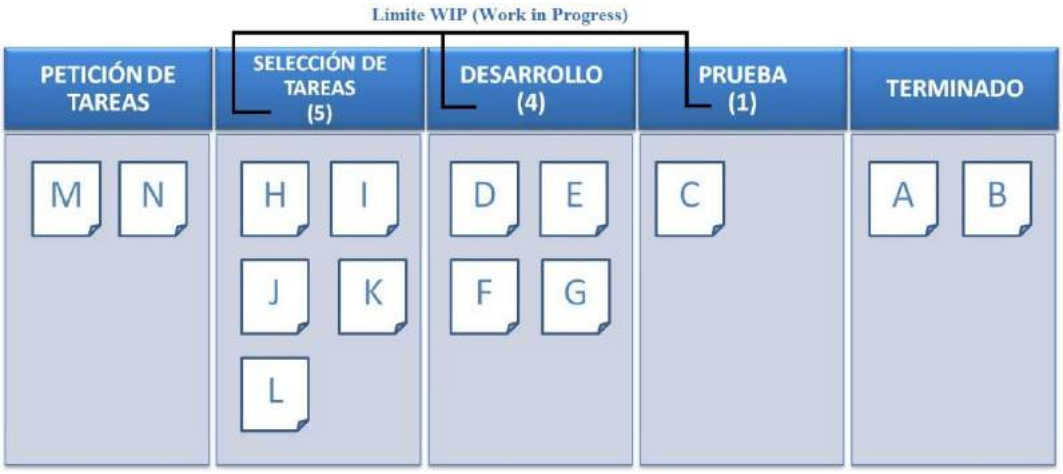


Ilustración 2-7: Muro Kanban

El objetivo de esta visualización es que quede claro el trabajo a realizar, en qué está trabajando cada persona, que todo el mundo tenga algo que hacer y el tener clara la prioridad de las tareas.

Las fases del ciclo de producción o flujo de trabajo se deben decidir según el caso, no hay nada acotado. En la figura se han puesto un conjunto de fases de ejemplo.

Determinar el límite del trabajo en curso (*Work In Progress*)

Quizás una de las principales ideas del Kanban es que el trabajo en curso (Work In Progress) debería estar limitado, es decir, que el número de tareas que se pueden realizar en cada fase debe ser algo conocido. Independientemente de si un proyecto es grande o

pequeño, simple o complejo, hay una cantidad de trabajo óptima que se puede realizar sin sacrificar eficiencia, por ejemplo, puede ser que realizar diez tareas a la vez nos lleve una semana, pero hacer dos cosas a la vez nos lleve sólo unas horas, lo que nos permite hacer quince tareas en la semana.

En Kanban se debe definir cuantas tareas como máximo puede realizarse en cada fase del ciclo de trabajo (ejemplo, como máximo 4 tareas en desarrollo, como máximo 1 en pruebas, etc.), a ese número de tareas se le llama límite del "work in progress". A esto se añade otra idea tan razonable como que para empezar con una nueva tarea alguna otra tarea previa debe haber finalizado.

Por ejemplo, en la anterior figura de ejemplo el número límite del "work in progress" para las pruebas es 1.

Medir el tiempo en completar una tarea (*Lead time*)

El tiempo que se tarda en terminar cada tarea se debe medir, a ese tiempo se le llama "lead time". El "lead time" cuenta desde que se hace una petición hasta que se hace la entrega

Aunque la métrica más conocida del Kanban es el "lead time", normalmente se suele utilizar también otra métrica importante: el "cycle time". El "cycle time" mide desde que el trabajo sobre una tarea comienza hasta que termina. Si con el "lead time" se mide lo que ven los clientes, lo que esperan, y con el "cycle time" se mide más el rendimiento del proceso.

Puede haber más métricas, pero las anteriores son las realmente importantes y necesarias para el control y mejora continua.

Roles

La metodología Kanban no prescribe roles. Tener un papel asignado y las tareas asociadas a dicho papel crean una identidad en el individuo. Por lo tanto, pedir que adopten un nuevo papel o un nuevo puesto de trabajo puede ser entendido como un ataque a su identidad. Habría una resistencia al cambio. Kanban trata de evitar esa resistencia emocional, entiende que la ausencia de papeles es una ventaja para el equipo.

Scrumban

Scrumban es una metodología derivada de los métodos de desarrollo Scrum y Kanban.

De Scrum

- Roles: Cliente, equipo (con los diferentes perfiles que se necesiten).
- Reuniones: reunión diaria.
- Herramientas: pizarra

De Kanban

- Flujo visual
- Hacer lo que sea necesario, cuando sea necesario y solo la cantidad necesaria.
- Limitar la cantidad de trabajo (WIP)
- Optimización del proceso.

Scrum vs Scrumban:

Normas	Scrum	Scrumban
Pizarra / Herramientas	Pizarra Backlogs Gráfica burn-down	Pizarra
Reuniones	Reunión diaria Planificación Retrospectiva	Reunión diaria
Iteraciones	Sí, Sprints	No, flujo continuo
Estimaciones	Sí	No
Esquipo	Multidisciplinar	Puede ser especializado
Roles	Product Owner Scrum Master Equipo	Equipo + otros
WIP (Work In Progress)	Controlado por el contenido del Sprint	Controlado por el estado de la tarea.
Cambios	Se pasan al siguiente Sprint	Se añaden al tablero en la columna “TO DO”.
Impedimentos	Solución inmediata	Se evitan.

Tabla 2-1: Scrum vs Scrumban

Es un modelo de desarrollo especialmente adecuado para proyectos de mantenimiento o proyectos en los que las historias de usuarios (requisitos del software) varíen con frecuencia o en los cuales surjan errores de programación inesperados durante todo el ciclo de desarrollo del producto. Para estos casos, los sprints (periodos de duración constante en los cuales se lleva a cabo un trabajo en sí) de la metodología Scrum no son factibles, dado que los errores/impedimentos que surgirán a lo largo de las tareas son difíciles de determinar y por lo tanto, no es posible estimar el tiempo que conlleva cada

historia. Por ello, resulta más beneficioso adoptar flujo de trabajo continuo propio del modelo Kanban.

Aunque hay diferencias entre ambos métodos, por ejemplo, las reglas de Kanban son muchas menos que las de Scrum, Kanban no define iteraciones (Sprints), Kanban limita explícitamente las tareas que se pueden realizar por fase (con el límite del work in progress), mientras que Scrum lo hace de manera indirecta por medio del sprint planning, etc.

Capítulo 3: Método comparativo

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Contenido

Método comparativo

El propósito de este apartado es responder a preguntas como:

¿Qué diferencia hay entre Srum y Kanban?

¿Dónde se complementan?

¿Hay conflictos potenciales?

En definitiva, este apartado trata de ser un comparador de herramientas, no se trata de juzgar cuál es mejor o cuál es peor.

Cuchillo o tenedor ¿qué herramienta es mejor?

Ilustración 3-1: ¿Qué herramienta es mejor?

Pues depende del contexto, para comer las albóndigas el tenedor probablemente sea mejor, para cortar setas el cuchillo y para comer filete lo mejor será utilizar ambos de manera conjunta.

Para la selección e implantación de una metodología existe una importante labor de documentación previa y, a partir de ahí, escoger alguna de las metodologías vistas para aplicar en el día a día de nuestro trabajo.

En este caso, se ha dado la vuelta al proceso, de manera que conociendo el marco de trabajo lleguemos a una metodología de trabajo apropiada.

Se ha elaborado un cuestionario que consta de dos partes, una inicial para conocer la orientación de la organización: ágil o tradicional y una segunda parte que permite conocer la metodología ágil que mejor se adapta al marco de trabajo de una organización.

Orientación de la organización

En esta fase se extraerá el enfoque de la organización, bien un enfoque tradicional o un enfoque ágil. Si la empresa sigue en un porcentaje alto de implantación de las directrices de las metodologías ágiles, se podría pasar a la segunda parte del formulario, mientras que si en esta primera fase se detecta que la cultura de trabajo es más cercana a una metodología tradicional, sería necesario conocer y adquirir las prácticas de una metodología ágil.

Primer formulario: Orientación tradicional vs orientación ágil

Para obtener este dato de forma objetiva, se analizará cada valor ágil y su relación con la organización.

Se han desglosado los valores del manifiesto ágil y se han dividido entre orientación ágil vs orientación tradicional, estos valores serán evaluados por la organización según una escala de importancia.

Valores de importancia:

0: Ninguna.

1: Baja importancia.

2: Media importancia.

3: Alta importancia.

ORIENTACIÓN ÁGIL		ORIENTACIÓN TRADICIONAL	
VALOR	**IMPORTANCIA**	**VALOR**	**IMPORTANCIA**
Individuo y las interacciones del equipo	*x1*	El proceso y las herramientas	*y1*
Desarrollar software que funciona	*x2*	Conseguir una buena documentación	*y2*
Colaboración con el cliente	*x3*	Negociación contractual	*y3*
Respuesta al cambio	*x4*	Seguimiento de un plan	*y4*

Tabla 3-1: Orientación tradicional vs orientación ágil

Siendo *x1, x2, x3 y x4* los valores asignados a cada valor con enfoque ágil, e *y1, y2, y3 e y4* los valores asignados a cada valor con enfoque tradicional.

Caso Práctico

ORIENTACIÓN ÁGIL		ORIENTACIÓN TRADICIONAL	
VALOR	**IMPORTANCIA**	**VALOR**	**IMPORTANCIA**
Individuo y las interacciones del equipo	3	El proceso y las herramientas	2
Desarrollar software que funciona	3	Conseguir una buena documentación	2
Colaboración con el cliente	2	Negociación contractual	2
Respuesta al cambio	3	Seguimiento de un plan	2
MEDIA	**2,75**		**2**

Tabla 3-2: Resultado orientación tradicional vs orientación ágil

En este caso, se demuestra que se sobrevalora lo indicado por los valores del manifiesto ágil.

Segundo Formulario: Cumplimiento principios ágiles

Se ha extraído a un formulario cada principio ágil y extraerá su relación con la organización.

Valores en prioridad:

0: Ninguna.

1: Baja prioridad.

2: Media prioridad.

3: Alta prioridad.

	Principios del Manifiesto Ágil	**Prioridad**
1	La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.	
2	Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.	
3	Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.	
4	La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.	

5	Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo	
6	El diálogo cara a cara es el método más efectivo para comunicar información dentro de un equipo de desarrollo.	
7	El software que funciona es la medida principal de progreso.	
8	Los procesos ágiles promueven un desarrollo sostenible. Los promotores, los desarrolladores y usuarios deberían ser capaces de mantener una paz constante	
9	La atención continua a la calidad técnica y al buen diseño mejora la agilidad.	
10	La simplicidad es esencial.	
11	Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.	
12	En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.	
	Total	**∑ prioridades asignadas**

Tabla 3-3: Cumplimiento principios ágiles

Siendo los principios ágiles 12 y la prioridad más alta tiene un valor de 3, entonces el valor más alto en cuanto al cumplimiento de estos principios de 36.

Según el resultado obtenido *Σ prioridades asignadas*, se deduce que las metas según el enfoque de la gerencia de sistemas de la empresa se orientan en un ***Σ prioridades asignadas* * 100/36 %** al cumplimiento de los principios ágiles.

Caso práctico

Indicadores de los principios ágiles en una organización según un desarrollador de la organización:

	Principios del Manifiesto Ágil	**Prioridad**
1	La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.	2
2	Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.	2
3	Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.	2
4	La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.	2
5	Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo	3
6	El diálogo cara a cara es el método más efectivo para comunicar información dentro de un equipo de desarrollo.	3

7	El software que funciona es la medida principal de progreso.	3
8	Los procesos ágiles promueven un desarrollo sostenible. Los promotores, los desarrolladores y usuarios deberían ser capaces de mantener una paz constante	3
9	La atención continua a la calidad técnica y al buen diseño mejora la agilidad.	3
10	La simplicidad es esencial.	3
11	Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.	2
12	En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.	2
	Total	**30**

Tabla 3-4: Ejemplo cumplimiento principios ágiles

Siendo los principios ágiles 12 y la prioridad más alta tiene un valor de 3, entonces el valor más alto en cuanto al cumplimiento de estos principios de 36.

Según el resultado obtenido de 30, se deduce que las metas a seguir de la organización según el desarrollador se orientan en un **83%** al cumplimiento integro o total de los principios ágiles.

El resultado será más representativo cuantos más miembros de la empresa realicen el cuestionario.

En el caso práctico que se acompaña se han obtenido datos objetivos que indican que la organización tiene un enfoque ágil, por tanto, el siguiente paso sería conocer qué metodología ágil, de entre las cuatro que se están evaluando en este Trabajo Fin de Grado, se adapta mejor a la organización que se está estudiando.

Elección de una metodología ágil

En este paso se evalúa la forma de trabajo de la empresa basándose en los cuatro puntos de vista de Iacovelli[1]. Para ello, se ha elaborado un nuevo formulario agrupando estos cuatro puntos: Uso, capacidad de agilidad, aplicación, procesos y productos. Cada uno de ellos con sus respectos atributos, cuyos valores serán asignados por la empresa en evaluación.

El objetivo del estudio realizado por Iacovelli: "Framework para la clasificación de metodologías ágiles", es clasificar los métodos a través de cuatro puntos de vista, cada uno representando un aspecto de las metodologías. Cada punto de vista se caracteriza por un conjunto de atributos.

[1] Adrian Iacovelli, autor de "Framework for Agile Methods Classification".

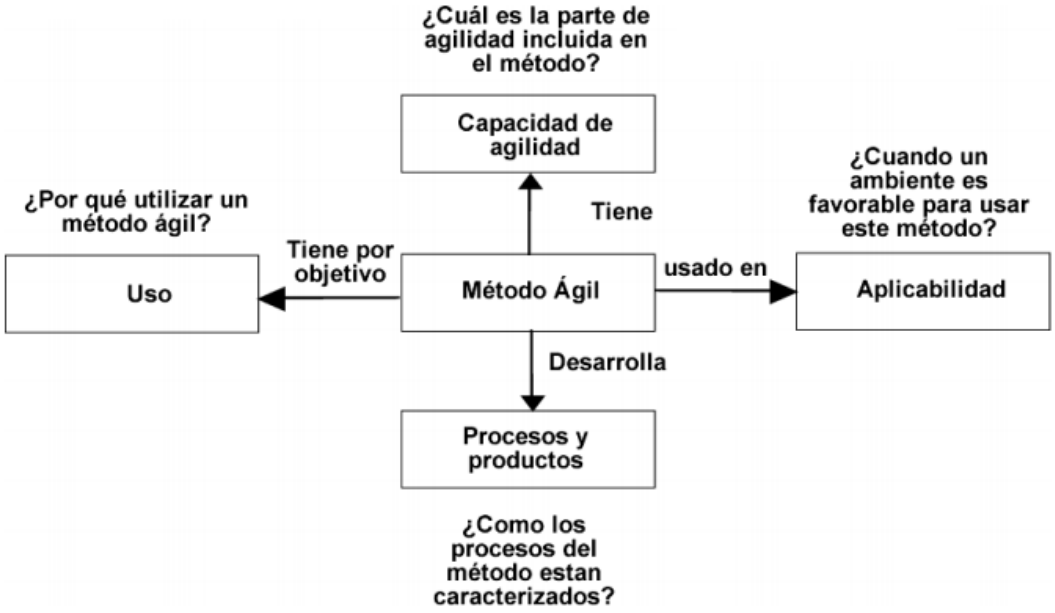


Ilustración 3-2: Cuatro vistas de las metodologías ágiles (Iacovelli)

USO

Refleja por qué utilizar metodologías ágiles. Los atributos de esta vista tratan de evaluar todos los beneficios de que el equipo de desarrollo y el cliente obtienen utilizando este tipo de metodologías: incremento de la productividad, calidad y satisfacción.

Las metodologías ágiles integran los cambios en el proceso de desarrollo, aportan reglas y directrices para trabajar en proyectos con requisitos cambiantes manteniendo fechas de entrega,… Las metodologías ágiles aportan flexibilidad.

Los atributos de este punto de vista son:

- Adaptarse a los entornos turbulentos.
- Satisfacción del usuario final.
- Favorable al offshoring (outsourcing internacional).
- Aumento de la productividad.
- El respeto de un nivel de calidad.
- El respeto de las fechas de entrega.
- Cumplimiento de los requisitos.

CAPACIDAD DE AGILIDAD

Representa cuál es la parte ágil de la metodología. Los atributos de esta vista representan todos los aspectos del concepto de agilidad y su evaluación refleja que aspectos están incluidos en una metodología.

Una metodología de desarrollo de software está compuesta por un ciclo de vida. En Ingeniería del Software existen diferentes ciclos de vida, como por ejemplo modelo en V, modelo en espiral,…. De acuerdo a la cronología de aparición de las metodologías ágiles, la mayoría de las metodologías derivan directamente del modelo en espiral. Esto se explica ya que las dos principales características de modelo en espiral son un ciclo de vida iterativo e incremental. Por tanto, los cambios de requisitos se pueden ir integrando en cada iteración, de manera que el plan de trabajo no tiene que ser modificado, irá cambiando a lo largo de las iteraciones. Otro punto interesante es la duración de las iteraciones, con iteraciones cortas aumenta el número de reuniones con el cliente para definir y detallar sus necesidades de forma incremental.

En cuanto a la interacción de las personas, las metodologías ágiles tienden a romper las relaciones contractuales entre los clientes y los equipos de desarrollo. Esta relación se expresa en este punto de vista por el atributo de colaboración. Un equipo ágil es un tipo de organización holográfica en la que cada miembro tiene el conocimiento del sistema en su conjunto, así que, si un miembro deja el equipo, no se ha perdido conocimiento.

El principal concepto de la agilidad son los procesos ligeros. Generalmente, las metodologías ágiles incluyen menos documentación. Las pruebas son una práctica muy importante, así como la refactorización.

Los atributos de este punto de vista son:

- Indicadores de cambio.
- Colaboración.
- Los requisitos funcionales pueden cambiar.
- Los recursos humanos pueden cambiar.
- Integración de los cambios.
- Nivel de intercambio de conocimientos (baja, alta).
- De peso ligero.
- Requisito no funcional puede cambiar.
- Centrado en las personas.
- Reactividad (al comienzo del proyecto, cada etapa, cada iteración).
- Refactoring político.
- Iteraciones cortas.
- Pruebas de política.
- Plan de trabajo se puede cambiar.

APLICABILIDAD

El objetivo de esta vista es mostrar el impacto de los aspectos ambientales en el método. Representa cuando el entorno es favorable para la aplicación de metodologías ágiles. Este aspecto se describe por atributos, cada uno correspondiente a una característica del entorno.

Los atributos de este punto de vista son:

- Grado de interacción entre los miembros del equipo (baja, alta).
- El grado de interacción con el cliente (baja, alta).
- Grado de interacción con los usuarios finales (baja, alta).
- Grado de integración de la novedad (baja, alta).
- La complejidad del proyecto (baja, alta).
- Los riesgos del proyecto (baja, alta).
- Tamaño del proyecto (pequeño, grande).
- La organización del equipo (auto-organización, la organización jerárquica).
- El tamaño del equipo (pequeño, grande).

PROCESOS Y PRODUCTOS

La vista de los procesos y productos representa cómo se caracteriza la metodología. Los atributos caracterizarán a los procesos ágiles por dos dimensiones y listarán los productos de las actividades del proceso.

El proceso se compone de dos dimensiones. La primera dimensión son las actividades de desarrollo de software cubiertas por las metodologías ágiles. La segunda representa el nivel de abstracción de sus directrices y reglas. Estas dos dimensiones se evalúan con atributos de esta vista.

Los atributos de los procesos y los productos son:

Nivel de abstracción de las normas y directrices:

- Gestión de proyectos
- Descripción de procesos
- Normas y orientaciones concretas sobre las actividades y productos

Las actividades cubiertas por el método ágil:

- Puesta en marcha del proyecto
- Definición de requisitos
- Modelado
- Código
- Pruebas unitarias
- Pruebas de integración
- Prueba del sistema
- Prueba de aceptación
- Control de calidad
- Sistema de uso

Productos de las actividades del método:

- Modelos de diseño
- Comentario del código fuente
- Ejecutable
- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación
- Informes de calidad
- Documentación de usuario

Tercer Formulario: Elección de una metodología ágil

USO

Verdadero (V) o Falso (F) en cada una de las premisas.

	Premisa	**Respuesta**
1	Respeto de las fechas de entrega	
2	Cumplimiento de los requisitos	
3	Respeto al nivel de calidad	
4	Satisfacción del usuario final	
5	Entornos turbulentos	
6	Favorable al Off shoring	
7	Aumento de la productividad	

Tabla 3-5: Valoración del Uso

CAPACIDAD DE AGILIDAD

Verdadero (V) o Falso (F) en cada una de las premisas.

	Premisa	**Respuesta**
1	Iteraciones cortas	
2	Colaboración	
3	Centrado en las personas	
4	Refactoring político	
5	Prueba político	
6	Integración de los cambios	
7	De peso ligero	
8	Los requisitos funcionales pueden cambiar	

9	Los requisitos no funcionales pueden cambiar	
10	El plan de trabajo puede cambiar	
11	Los recursos humanos pueden cambiar	
12	Cambiar los indicadores	
13	Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACIÓN)	
14	Intercambio de conocimientos (BAJO, ALTO)	

Tabla 3-6: Valoración de la Capacidad de agilidad

APLICACIÓN

Verdadero (V) o Falso (F) en cada una de las premisas.

	Premisa	**Respuesta**
1	Tamaño del proyecto (PEQUEÑO, GRANDE)	
2	La complejidad del proyecto (BAJA, ALTA)	
3	Los riesgos del proyecto (BAJO, ALTO)	
4	El tamaño del equipo (PEQUEÑO, GRANDE)	
5	El grado de interacción con el cliente (BAJA, ALTA)	
6	Grado de interacción con los usuarios finales (BAJA, ALTA)	
7	Grado de interacción entre los miembros del equipo (BAJA, ALTA)	
8	Grado de integración de la novedad (BAJA, ALTA)	
9	La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	

Tabla 3-7: Valoración de la Aplicación

PROCESOS Y PRODUCTOS

Verdadero (V) o Falso (F) en cada una de las premisas.

Nivel de abstracción de las normas y directrices:

	Premisa	**Respuesta**
1	Gestión de proyectos	
2	Descripción de procesos	
3	Normas y orientaciones concretas sobre las actividades y productos	

Tabla 3-8: Valoración normas y directrices ágiles

Las actividades cubiertas por el método ágil:

	Premisa	**Respuesta**
1	Puesta en marcha del proyecto	
2	Definición de requisitos	
3	Modelado	
4	Código	

5	Pruebas unitarias	
6	Pruebas de integración	
7	Prueba del sistema	
8	Prueba de aceptación	
9	Control de calidad	
10	Sistema de uso	

Tabla 3-9: Valoración actividades cubiertas por el método ágil

Productos de las actividades del método:

	Premisa	**Respuesta**
1	Modelos de diseño	
2	Comentario del código fuente	
3	Ejecutable	
4	Pruebas unitarias	
5	Pruebas de integración	
6	Pruebas de sistema	
7	Pruebas de aceptación	
8	Informes de calidad	
9	Documentación de usuario	

Tabla 3-10: Valoración de los productos de las actividades de la metodología ágil

Los datos extraídos de este formulario se compararán con clasificación de las diferentes metodologías que se muestra a continuación.

			METODOLOGÍAS ÁGILES			
			ORIENTADA AL DESARROLLO DE SOFTWARE	**ORIENTADA A LA GESTIÓN DE PROYECTOS**		
			XP	**SCRUM**	**KANBAN**	**SCRUMBAN**
USO	¿Por qué utilizar un método ágil?	Respeto de las fechas de entrega	FALSO	VERDADERO	FALSO	FALSO
		Cumplimiento de los requisitos	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Respeto al nivel de calidad	FALSO	FALSO	FALSO	FALSO
		Satisfacción del usuario final	FALSO	VERDADERO	FALSO	FALSO
		Entornos turbulentos	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Favorable al Off shoring	FALSO	VERDADERO	FALSO	VERDADERO
		Aumento de la productividad	VERDADERO	VERDADERO	VERDADERO	VERDADERO
CAPACIDAD DE AGILIDAD	¿Cuál es la parte de agilidad incluida en el método?	Iteraciones cortas	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Colaboración	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Centrado en las personas	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Refactoring político	VERDADERO	FALSO	FALSO	FALSO

		Prueba político	VERDADERO	VERDADERO	FALSO	VERDADERO
		Integración de los cambios	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		De peso ligero	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Los requisitos funcionales pueden cambiar	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Los requisitos no funcionales pueden cambiar	FALSO	FALSO	VERDADERO	VERDADERO
		El plan de trabajo puede cambiar	VERDADERO	FALSO	VERDADERO	VERDADERO
		Los recursos humanos pueden cambiar	VERDADERO	FALSO	VERDADERO	VERDADERO
		Cambiar los indicadores	VERDADERO	FALSO	FALSO	FALSO
		Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACIÓN)	ITERACIÓN	ITERACIÓN	ITERACIÓN	ITERACIÓN
		Intercambio de conocimientos (BAJO, ALTO)	ALTO	BAJO	BAJO	BAJO
APLICABILIDAD	¿Cuándo un ambiente es favorable para usar este método?	Tamaño del proyecto (PEQUEÑO, GRANDE)	PEQUEÑO	GRANDE / PEQUEÑO	PEQUEÑO	GRANDE / PEQUEÑO
		La complejidad del proyecto (BAJA, ALTA)	BAJA	ALTA	BAJA	ALTA
		Los riesgos del proyecto (BAJO, ALTO)	BAJO	ALTO	BAJO	ALTO
		El tamaño del equipo (PEQUEÑO, GRANDE)	PEQUEÑO	PEQUEÑO	PEQUEÑO	PEQUEÑO
		El grado de interacción con el cliente (BAJA, ALTA)	ALTA	ALTA	BAJO	BAJA
		Grado de interacción con los usuarios finales (BAJA, ALTA)	BAJO	ALTA	BAJO	BAJA
		Grado de interacción entre los miembros del equipo (BAJA, ALTA)	ALTA	ALTA	BAJA	ALTA
		Grado de integración de la novedad (BAJA, ALTA)	ALTA	ALTA	BAJA	ALTA
		La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN
PROCESOS Y PRODUCTOS	¿Cómo están caracterizados los procesos del método?	**Nivel de abstracción de las normas y directrices**				
		Gestión de proyectos	FALSO	VERDADERO	FALSO	VERDADERO
		Descripción de procesos	VERDADERO	FALSO	FALSO	FALSO
		Normas y orientaciones concretas sobre las actividades y productos	VERDADERO	FALSO	FALSO	FALSO
		Las actividades cubiertas por el método ágil				

		Puesta en marcha del proyecto	FALSO	FALSO	FALSO	FALSO
		Definición de requisitos	VERDADERO	VERDADERO	FALSO	VERDADERO
		Modelado	VERDADERO	VERDADERO	FALSO	FALSO
		Código	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Pruebas unitarias	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Pruebas de integración	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Prueba del sistema	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Prueba de aceptación	FALSO	FALSO	FALSO	FALSO
		Control de calidad	FALSO	FALSO	FALSO	FALSO
		Sistema de uso	FALSO	FALSO	FALSO	FALSO
		Productos de las actividades del método ágil				
		Modelos de diseño	FALSO	VERDADERO	FALSO	VERDADERO
		Comentario del código fuente	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Ejecutable	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Pruebas unitarias	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Pruebas de integración	VERDADERO	VERDADERO	VERDADERO	VERDADERO
		Pruebas de sistema	VERDADERO	FALSO	VERDADERO	VERDADERO
		Pruebas de aceptación	FALSO	FALSO	FALSO	FALSO
		Informes de calidad	FALSO	FALSO	FALSO	FALSO
		Documentación de usuario	FALSO	FALSO	FALSO	FALSO

Tabla 3-11: Clasificación de metodologías ágiles

Comparando los resultados obtenidos del formulario y la tabla de clasificación anterior, se identificará la metodología que mejor se adapta a la forma de trabajo de la empresa. La metodología adecuada será la que mayor número de coincidencias tenga con el cuestionario anterior.

Caso práctico

USO

	Premisa	**Respuesta**
1	Respeto de las fechas de entrega	V
2	Cumplimiento de los requisitos	V
3	Respeto al nivel de calidad	V
4	Satisfacción del usuario final	V
5	Entornos turbulentos	V
6	Favorable al Off shoring	F
7	Aumento de la productividad	V

Tabla 3-12: Ejemplo valoración del Uso

CAPACIDAD DE AGILIDAD

	Premisa	**Respuesta**
1	Iteraciones cortas	V
2	Colaboración	V
3	Centrado en las personas	V
4	Refactoring político	F
5	Prueba político	V
6	Integración de los cambios	V
7	De peso ligero	V
8	Los requisitos funcionales pueden cambiar	V
9	Los requisitos no funcionales pueden cambiar	V
10	El plan de trabajo puede cambiar	V
11	Los recursos humanos pueden cambiar	V
12	Cambiar los indicadores	V
13	Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACIÓN)	ITERACIÓN
14	Intercambio de conocimientos (BAJO, ALTO)	BAJO

Tabla 3-13: Ejemplo valoración de la Capacidad de agilidad

APLICACIÓN

	Premisa	**Respuesta**
1	Tamaño del proyecto (PEQUEÑO, GRANDE)	PEQUEÑO
2	La complejidad del proyecto (BAJA, ALTA)	BAJA
3	Los riesgos del proyecto (BAJO, ALTO)	BAJO
4	El tamaño del equipo (PEQUEÑO, GRANDE)	PEQUEÑO
5	El grado de interacción con el cliente (BAJA, ALTA)	ALTA
6	Grado de interacción con los usuarios finales (BAJA, ALTA)	ALTA
7	Grado de interacción entre los miembros del equipo (BAJA, ALTA)	ALTA
8	Grado de integración de la novedad (BAJA, ALTA)	ALTA
9	La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	JERÁRQUICA

Tabla 3-14: Ejemplo valoración de la Aplicación

PROCESOS Y PRODUCTOS

Verdadero (V) o Falso (F) en cada una de las premisas.

Nivel de abstracción de las normas y directrices:

	Premisa	Respuesta
1	Gestión de proyectos	V
2	Descripción de procesos	V
3	Normas y orientaciones concretas sobre las actividades y productos	F

Tabla 3-15: Ejemplo valoración de las normas y directrices de la metodología ágil

Las actividades cubiertas por el método ágil:

	Premisa	Respuesta
1	Puesta en marcha del proyecto	V
2	Definición de requisitos	V
3	Modelado	F
4	Código	V
5	Pruebas unitarias	V
6	Pruebas de integración	V
7	Prueba del sistema	V
8	Prueba de aceptación	V
9	Control de calidad	V
10	Sistema de uso	V

Tabla 3-16: Ejemplo valoración de las actividades cubiertas por la metodología ágil

Productos de las actividades del método:

	Premisa	Respuesta
1	Modelos de diseño	F
2	Comentario del código fuente	V
3	Ejecutable	V
4	Pruebas unitarias	V
5	Pruebas de integración	V
6	Pruebas de sistema	V
7	Pruebas de aceptación	V
8	Informes de calidad	V
9	Documentación de usuario	V

Tabla 3-17: Ejemplo valoración de los productos de las actividades de la metodología ágil

En los casos en los que la respuesta de la organización coincida con el valor asociado a la metodología se sumará 1, en caso contrario 0.

		METODOLOGÍAS ÁGILES			
		ORIENTADA AL DESARROLLO DE SOFTWARE	ORIENTADA A LA GESTIÓN DE PROYECTOS		
		XP	SCRUM	KANBAN	SCRUMBAN
USO	Respeto de las fechas de entrega	0	1	0	0
	Cumplimiento de los requisitos	1	1	1	1
	Respeto al nivel de calidad	0	0	0	0
	Satisfacción del usuario final	0	1	0	0
	Entornos turbulentos	1	1	1	1
	Favorable al Off shoring	1	0	1	0
	Aumento de la productividad	1	1	1	1
CAPACIDAD DE AGILIDAD	Iteraciones cortas	1	1	1	1
	Colaboración	1	1	1	1
	Centrado en las personas	1	1	1	1
	Refactoring político	0	1	1	1
	Prueba político	1	1	0	1
	Integración de los cambios	1	1	1	1
	De peso ligero	1	1	1	1
	Los requisitos funcionales pueden cambiar	1	1	1	1
	Los requisitos no funcionales pueden cambiar	0	0	1	1
	El plan de trabajo puede cambiar	1	0	1	1
	Los recursos humanos pueden cambiar	1	0	1	1
	Cambiar los indicadores	1	0	0	0
	Reactividad	1	1	1	1
	Intercambio de conocimientos	0	1	1	1
APLICABILIDAD	Tamaño del proyecto	1	1	1	1
	La complejidad del proyecto	1	0	1	0
	Los riesgos del proyecto	1	0	1	0
	El tamaño del equipo	1	1	1	1
	El grado de interacción con el cliente	1	1	0	0
	Grado de interacción con los usuarios finales	0	1	0	0
	Grado de interacción entre los miembros del equipo	1	1	0	1
	Grado de integración de la novedad	1	1	0	1

	La organización del equipo	0	0	0	0
PROCESOS Y PRODUCTOS	**Nivel de abstracción de las normas y directrices**				
	Gestión de proyectos	0	1	0	1
	Descripción de procesos	1	0	0	0
	Normas y orientaciones concretas sobre las actividades y productos	0	1	1	1
	Las actividades cubiertas por el método ágil				
	Puesta en marcha del proyecto	0	0	0	0
	Definición de requisitos	1	1	0	1
	Modelado	0	0	1	1
	Código	1	1	1	1
	Pruebas unitarias	1	1	1	1
	Pruebas de integración	1	1	1	1
	Prueba del sistema	1	1	1	1
	Prueba de aceptación	0	0	0	0
	Control de calidad	0	0	0	0
	Sistema de uso	0	0	0	0
	Productos de las actividades del método ágil				
	Modelos de diseño	1	0	1	0
	Comentario del código fuente	1	1	1	1
	Ejecutable	1	1	1	1
	Pruebas unitarias	1	1	1	1
	Pruebas de integración	1	1	1	1
	Pruebas de sistema	1	0	1	1
	Pruebas de aceptación	0	0	0	0
	Informes de calidad	0	0	0	0
	Documentación de usuario	0	0	0	0
	TOTAL:	**34**	32	32	**33**

Tabla 3-18: Metodología ágil adecuada para el ejemplo

En los resultados se observa que XP y SCRUMBAN son los que han obtenido más puntuación. Podrían aplicar una u otra, incluso XP para la fase de desarrollo y SCRUM para la gestión de los proyectos de la organización.

Conclusiones

Usar las herramientas adecuadas ayuda a triunfar, pero no garantiza el éxito. Es fácil confundir el éxito/fracaso del proyecto, con el éxito/fracaso de la herramienta.

Del mismo modo, el hecho de no aplicar todas y cada una de las recomendaciones de la herramienta no conduce al fracaso. No es requisito indispensable para el éxito.

- Un proyecto puede triunfar debido a una gran herramienta.
- Un proyecto puede triunfar a pesar de una pésima herramienta.
- Un proyecto puede fallar debido a una pésima herramienta.
- Un proyecto puede fallar a pesar de una gran herramienta.

No limitar las herramientas que utilizamos, las herramientas (Scrum, Kanban, XP) se pueden combinar, por ejemplo, muchos equipos de Kanban hacen reuniones diarias (que es una práctica de Scrum). Hay que usar lo que funcione en cada equipo.

Capítulo 4: Caso Práctico

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Contenido

Caso práctico

En este capítulo se muestra un caso práctico de una iteración (Sprint) de Scrum para resolver una seria de tareas para un proyecto Web. El objetivo es tener una toma de contacto con una aplicación práctica de las metodologías ágiles y no solo teórica. A lo largo del capítulo se han ido documentando el día a día de la iteración.

Velocidad inicial del equipo

Antes de comenzar con la reunión de planificación se calculan el número de puntos ideales de trabajo que es capaz de asumir el equipo en el presente Sprint. Es necesario tener en cuentas los siguientes parámetros:

Duración del Sprint: 15 días laborables (3 semanas).

Número de desarrolladores del equipo y su disponibilidad: 5 desarrolladores.

Factor de corrección: 0,6

Se calcula el número de días de trabajo teóricos que es capaz de asumir el equipo:

	Disponibilidad	Días de trabajo en el Sprint
Desarrollador 1	100%	15
Desarrollador 2	100%	15
Desarrollador 3	100%	15
Desarrollador 4	100%	15
Desarrollador 5	50%	7,5
TOTAL		**67,5**

días de trabajo teóricos

Tabla 4-1: Velocidad del equipo

Teniendo en cuenta el factor de corrección anterior, se calculan los días reales de trabajo.

67,5 días de trabajo teóricos * 0,6 = 40.5

El equipo es capaz de asumir **40.5** puntos de trabajo en el Sprint.

Reunión de planificación

Backlog de historias de usuario presentado por el cliente:

Prior	Proyecto	Nombre	Notas	Cómo probarlo
1	Gestión Incidencias	Crear tarea para que procese el excel y lo transforme en incidencias en BBDD.	Script que procese los excel subidos por los usuarios y los transforme en incidencias.	Comprobar que una vez que el usuario sube un Excel con incidencias, se procesan y se crean registros de incidencias en la web.
2	Gestión Incidencias	Crear back-end para subir Excel con incidencias	Almacenar el fichero Excel en BBDD.	
3	Gestión Incidencias	Crear front-end para subir Excel con incidencias.	Nueva opción en el menú y nueva ventana para que el usuario descargue la plantilla Excel y una vez cubierta la envíe al sistema.	Comprobar guía de estilo del cliente. Test de accesibilidad.
4	LDAP	Separar vista para Administradores / No Administradores.	Permisos según perfiles: - Administrador: Tendrá acceso a los informes de todos los distribuidores - No administrador: Solo acceso a sus propios informes	Comprobar que un usuario Administrador puede adoptar el rol de un distribuidor y consultar sus informes. Comprobar que un usuario no administrador solo tiene acceso a sus propios informes.
5	LDAP	Configuración de perfiles	Es necesario tener dos perfiles diferentes a la hora de acceder a la Web: Los perfiles a configurar son: - Administrador. - No administrador.	
6	LDAP	Instalar Servidor LDAP	Se requiere implantar LDAP en el acceso a la Web de Comisiones.	
7	Informes	Añadir un nuevo concepto de comisión en los informes	Se necesita el que concepto por el que se comisionan las portabilidades aparezca desglosado en una nueva columna de los informes de la Web de Comisiones. El nombre de la columna será: "Comisión Portabilidad" y se acumulará la cantidad comisionada al distribuidor en concepto de portabilidad durante el ciclo de comisiones.	Una vez calculada la comisión por portabilidad, comprobar que se acumula en la columna de "Comisión Portabilidad".

Durante la reunión de planificación se trató una a una y con la prioridad establecida por el cliente las historias de usuario para el siguiente Sprint.

Utilizando para la estimación la serie de Fibonacci (1/2, 1, 2, 3, 5, 6, 7, ∞, ).

El símbolo ∞ se muestra cuando un miembro del equipo es incapaz de estimar el tiempo que le llevaría una tarea, en ese caso el resto del equipo hace una presentación técnica de la tarea para que todo el mundo pueda estimar.

 , se muestra la taza de café cuando el equipo necesita cinco minutos de descanso. El tiempo de duración de una reunión de planificación puede ser como máximo de cuatro horas, por eso se contempla hacer descansos durante esta tarea.

Los resultados obtenidos de la reunión de planificación son las historias de usuario que se listan a continuación, que incluyen su estimación, tareas en las que se descompone y prioridad:

Historias de usuario

Backlog Item #7
Proyecto: Informes

Añadir un nuevo concepto de comisión en los informes

Notas	Prioridad
Se necesita el que concepto por el que se comisionan las portabilidades aparezca desglosado en una nueva columna de los informes de la Web de Comisiones. El nombre de la columna será: "Comisión Portabilidad" y se acumulará la cantidad comisionada al distribuidor en concepto de Portabilidad durante el ciclo de comisiones.	7
Como probarlo	**Estimación**
Una vez calculada la comisión por portabilidad, comprobar que se acumula en la columna de "Comisión Portabilidad".	11

Tareas asociadas a esta historia de usuario:

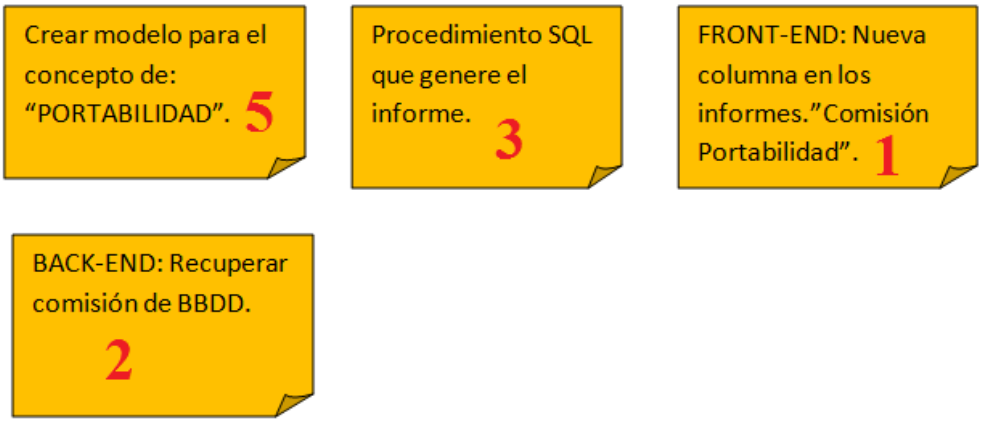


Ilustración 4-1: Caso Práctico – Tareas de Backlog tem 7

- Crear un modelo para el cálculo de este tipo de comisión asociado al concepto: "PORTABILIDAD".

- Crear procedimiento SQL que agrupe las comisiones de “PORTABILIDAD” por distribuidor y las incluya en el informe correspondiente
- Font-end: Añadir una nueva columna en la vista de los informes en la web para el concepto de “PORTABILIDAD”. El nombre de la columna será: “Comisión Portabilidad”.
- Back-end: Recuperar de BBDD los la cantidad de comisión por portabilidad para mostrarla en los informes.

Backlog Item #6
Proyecto: LDAP

Instalar Servidor LDAP

Notas

Se requiere implantar LDAP en el acceso a la Web de Comisiones.

Prioridad

6

Como probarlo

Estimación

3

Tareas asociadas a esta historia de usuario:

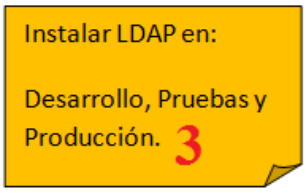


Ilustración 4-2: Caso Práctico – Tareas de Backlog tem 6

- Instalar el servidor de LDAP en los diferentes entornos: Desarrollo, Pruebas y Producción.

Backlog Item #5
Proyecto: LDAP

Configuración de perfiles

Notas

Es necesario tener dos perfiles diferentes a la hora de acceder a la Web:

Los perfiles a configurar son:
- Administrador.
- No administrador.

Prioridad

5

Como probarlo

Estimación

2

Tareas asociadas a esta historia de usuario:

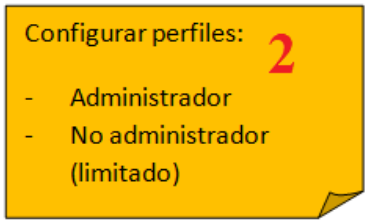


Ilustración 4-3: Caso Práctico – Tareas de Backlog tem 5

- Configurar usuario con perfil administrador.
- Configurar usuario con perfil no administrador (limitado).

Backlog Item #4
Proyecto: LDAP

Separar vista para Administradores / No Administradores.

Notas	Prioridad
Permisos según perfiles: - Administrador: Tendrá acceso a los informes de todos los distribuidores - No administrador: Solo acceso a sus propios informes	4
Como probarlo	**Estimación**
Comprobar que un usuario Administrador puede adoptar el rol de un distribuidor y consultar sus informes. Comprobar que un usuario no administrador solo tiene acceso a sus propios informes.	5

Tareas asociadas a esta historia de usuario:

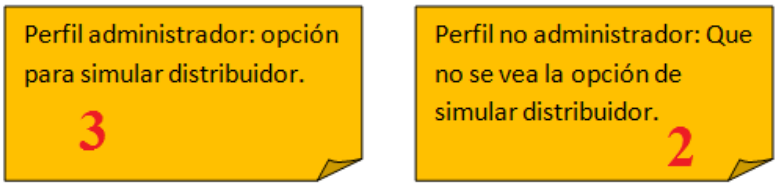


Ilustración 4-4: Caso Práctico – Tareas de Backlog tem 4

- Nueva opción para simular distribuidor en el perfil administrador.
- Perfil no administrador que no se vea esta opción.

Backlog Item #3
Proyecto: Gestión Incidencias

Crear front-end para subir Excel con incidencias.

Notas	Prioridad
Nueva opción en el menú y nueva ventana para que el usuario descargue la plantilla Excel y una vez cubierta la envíe al sistema.	3

Como probarlo	Estimación
Comprobar guía de estilo del cliente. Test de accesibilidad.	6

Tareas asociadas a esta historia de usuario:

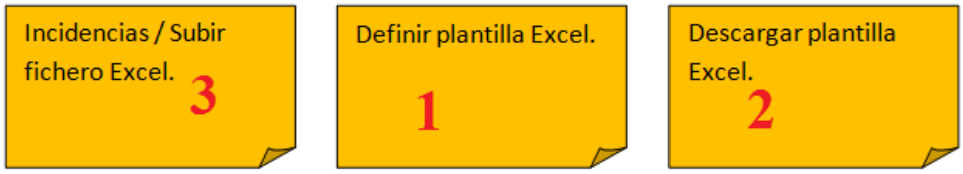


Ilustración 4-5: Caso Práctico – Tareas de Backlog tem 3

- En la opción de incidencias crear un apartado para subir fichero Excel.
- Definir plantilla Excel.
- En la opción de incidencias poner enlace para descargar plantilla Excel de incidencias.

Backlog Item #2
Proyecto: Gestión Incidencias

Crear back-end para subir Excel con incidencias.

Notas	Prioridad
Almacenar el fichero Excel en BBDD.	2
Como probarlo	**Estimación**
	4

Tareas asociadas a esta historia de usuario:

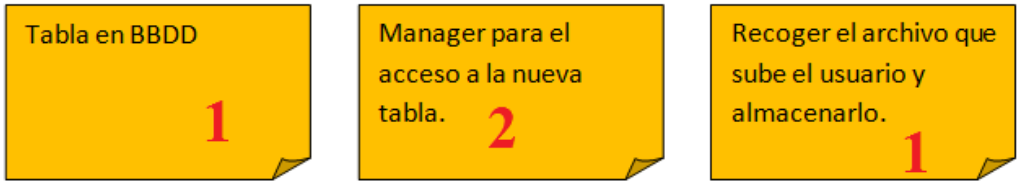


Ilustración 4-6: Caso Práctico – Tareas de Backlog tem 2

- Crear tabla de BBDD como repositorio de los archivos Excel con incidencias de los usuarios.
- Crear Manager para el acceso a la nueva tabla.
- Recoger el archivo Excel de la vista y almacenarlo.

Backlog Item #1
Proyecto: Gestión Incidencias

Crear tarea para que procese el Excel y lo transforme en incidencias en BBDD.

Notas	Prioridad
Script que procese los excel subidos por los usuarios y los transforme en incidencias.	1

Como probarlo	Estimación
Comprobar que una vez que el usuario sube un Excel con incidencias, se procesan y se crean registros de incidencias en la web.	9

Tareas asociadas a esta historia de usuario:

Ilustración 4-7: Caso Práctico – Tareas de Backlog tem 1

- Crear script Perl para que lea el fichero Excel.
- Crear incidencias de usuario e insertarlas en la tabla de incidencias.
- Añadir validaciones al script.
- Planificar Script en un crontab para que se ejecute cada hora.

La suma total de los puntos de cada historia de usuario es: 40

Si antes de comenzar la reunión de planificación se había calculado que la cantidad de trabajo que podía asumir el equipo era de 40.5 puntos, el equipo puede comprometerse a implementar las siete historias de usuario propuestas por el cliente para este Sprint.

Reuniones diarias

Inicio del Sprint

Los miembros del equipo se reúnen en frente al tablón de Scrum. En la primera reunión nadie tiene una tarea asignada, por tanto, no es necesario responder a las preguntas: ¿Qué has hecho ayer? ¿Qué voy a hacer hoy? ¿Algún impedimento?. Cada desarrollador coge una tarea del tablón y comienza con la implementación.

Tareas que ha cogido cada desarrollador:

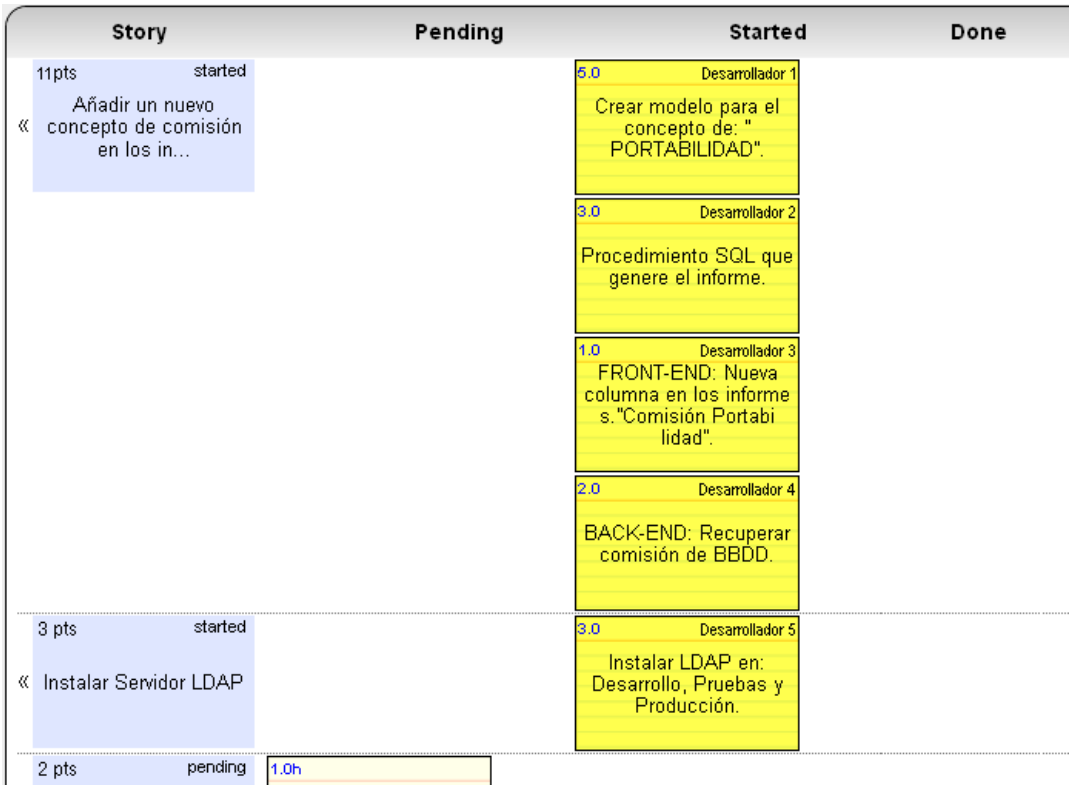


Ilustración 4-8: Caso Práctico - Tablón al inicio del Sprint

Estado de la gráfica

La línea gris marca el progreso ideal del Sprint, se pinta para tener una referencia de la consecución de trabajo a lo largo del Sprint, siempre que la línea de progreso vaya por

debajo de la línea gris, el ritmo es mejor del esperado, cuando está por encima, el equipo es más lento de lo planificado.

El primer día del Sprint la gráfica está limpia, aún no ha habido progreso en las tereas.

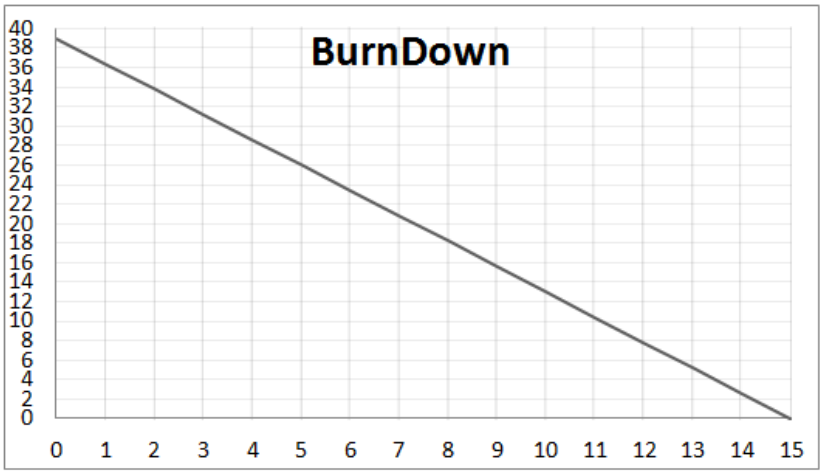


Ilustración 4-9: Caso Práctico - Burndown al inicio del sprint

Día 1:

Preguntas que tiene que responder cada uno de los desarrolladores: ¿Qué has hecho ayer? ¿Qué voy a hacer hoy? ¿Algún impedimento?

Desarrollador 1: Ha estado trabajando en la tarea: "Crear modelo para el concepto de PORTABILIDAD". Durante el día de hoy continuará con esta tarea. No ha encontrado ningún impedimento y estima que aún le quedan 4 puntos de trabajo, por tanto, ha avanzado un punto de trabajo.

Desarrollador 2: Ha estado trabajando en la tarea: "Procedimiento SQL que genere informe". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Estima que le quedan 2 puntos de trabajo en esta tarea.

Desarrollador 3: Ha estado trabajando en la tarea: "FRONT-END: Nueva columna en los informes. Comisión Portabilidad" y ya está terminada, por tanto, actualiza el ticket de la tarea a la columna "**¡Hecho!**". Coge la siguiente tarea disponible en el tablón: "Configurar perfil: administrador".

Desarrollador 4: Ha estado trabajando en la tarea: "BACK-END: Recuperar comisión de BBDD". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Estima que le quedan 1 puntos de trabajo en esta tarea.

Desarrollador 5: Ha estado trabajando en la tarea: "Instalar LDAP en: Desarrollo, Pruebas y Producción", continuará trabajando en esta tarea y la reestima en 2.5 puntos de trabajo.

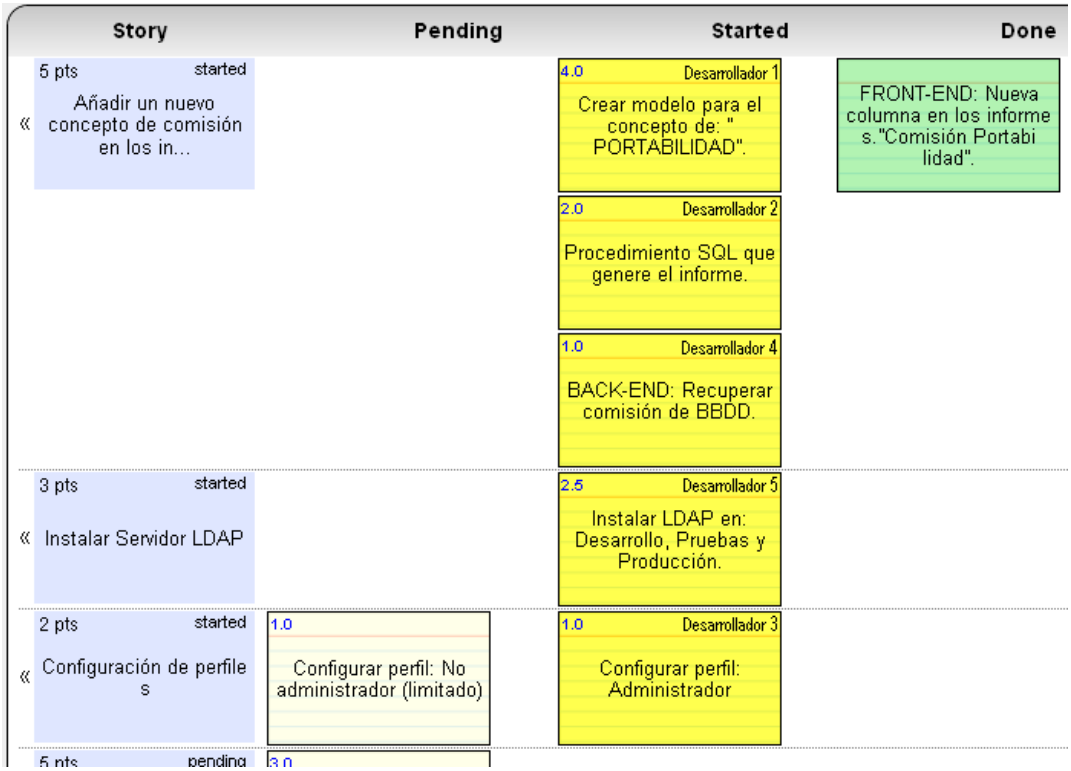


Ilustración 4-10: Caso Práctico - Tablón primer día del Sprint

Estado actualizado de la gráfica

Se han quitado 3,5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 35,5 puntos de trabajo restantes.

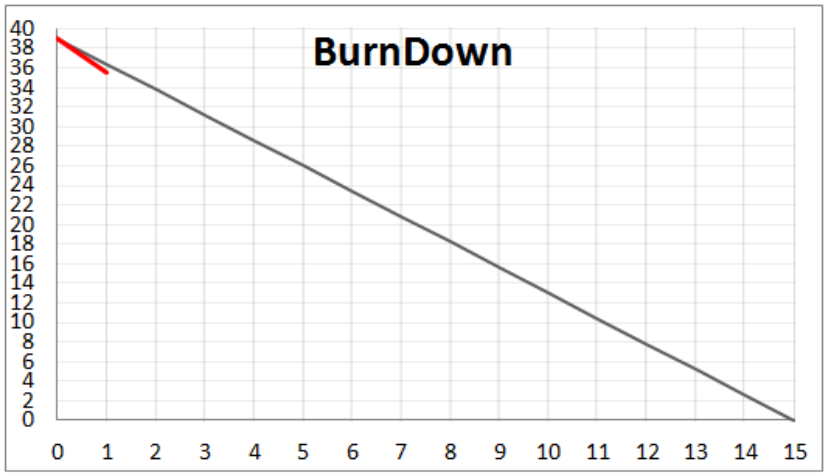


Ilustración 4-11: Caso Práctico - Burndown primer día del Sprint

Día 2:

Desarrollador 1: Ha estado trabajando en la tarea: "Crear modelo para el concepto de PORTABILIDAD". Durante el día de hoy continuará con esta tarea. No ha encontrado ningún impedimento y estima que aún le quedan 3 puntos de trabajo, por tanto, ha avanzado un punto de trabajo.

Desarrollador 2: Ha estado trabajando en la tarea: "Procedimiento SQL que genere informe". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Estima que le quedan 1.5 puntos de trabajo en esta tarea.

Desarrollador 3: Ha estado trabajando en la tarea: "Configurar perfil: administrador" en el entorno de Desarrollo. Mientras no se instalen los servidores en los entornos de Pruebas y Producción no puede seguir trabajando, por tanto, deja su tarea en la columna: "**Pendiente**" y coge la tarea de: "Instalar LDAP en: Pruebas y Producción".

Desarrollador 4: Ha estado trabajando en la tarea: "BACK-END: Recuperar comisión de BBDD". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Mantiene la estimación de la tarea.

Desarrollador 5: El desarrollador 5 (su disponibilidad al Sprint es del 50%) no ha podido trabajar en la tarea de: "Configurar perfil: Administrador", y durante la jornada de hoy tampoco podrá dedicarle tiempo, por tanto, mantiene la estimación de la tarea.

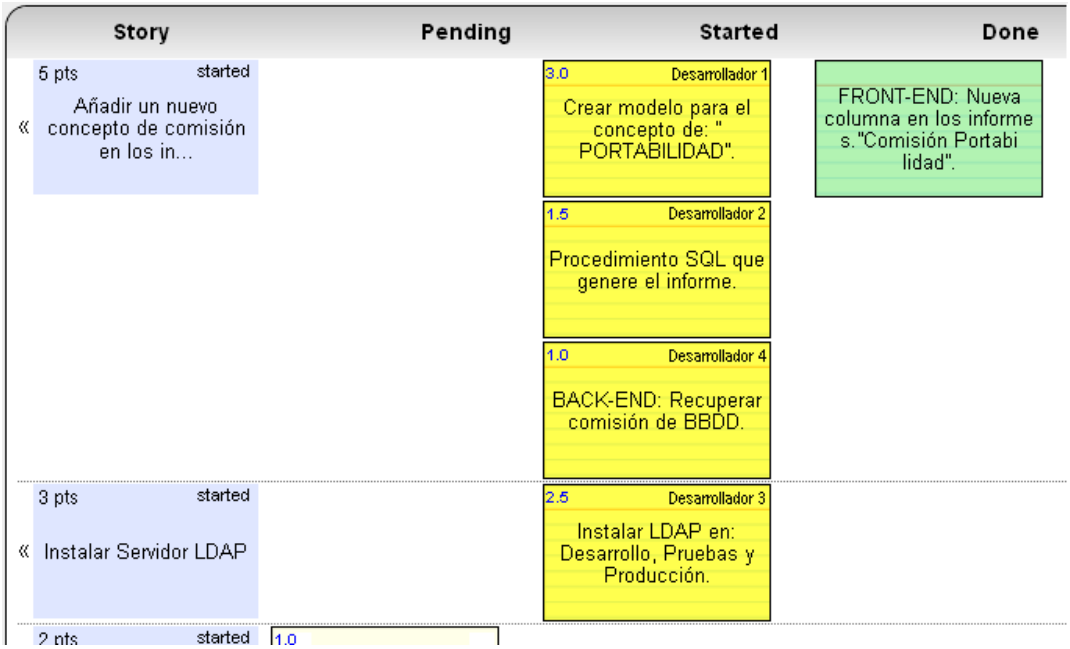


Ilustración 4-12: Caso Práctico - Tablón segundo día del Sprint

Estado actualizado de la gráfica

Se han quitado 1,5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 34 puntos de trabajo restantes.

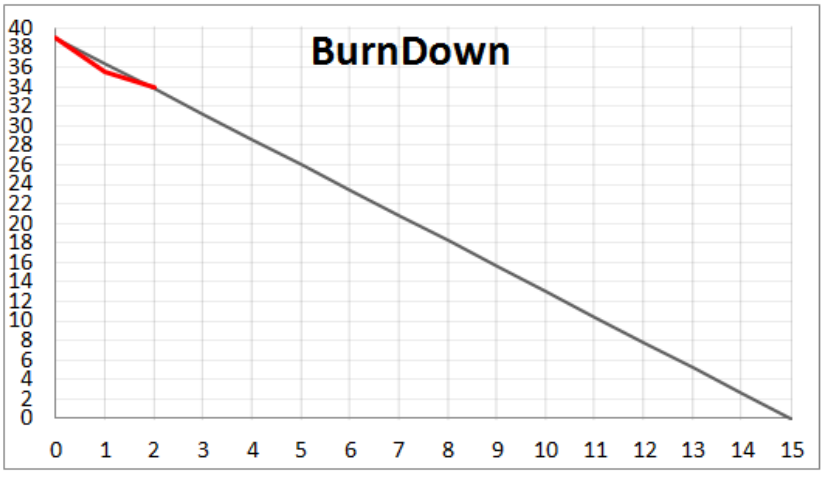


Ilustración 4-13: Caso Práctico - Burndown segundo día del Sprint

Día 3:

Desarrollador 1: Ha estado trabajando en la tarea: "Crear modelo para el concepto de PORTABILIDAD". Durante el día de hoy continuará con esta tarea. No ha encontrado ningún impedimento y estima que aún le quedan 2.5 puntos de trabajo.

Desarrollador 2: Ha estado trabajando en la tarea: "Procedimiento SQL que genere informe". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Estima que aún le quedan 1.5 puntos de trabajo en esta tarea.

Desarrollador 3: Ha finalizado la tarea: "Instalar LDAP en: Desarrollo, Pruebas y Producción" y la coloca en la columna "**¡Hecho!**". Para continuar hoy se coge la tarea: "Configurar perfil: Administrador".

Desarrollador 4: Ha estado trabajando en la tarea: "BACK-END: Recuperar comisión de BBDD". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Reestima la tarea en 0.5 puntos de trabajo.

Desarrollador 5: El desarrollador 5 (su disponibilidad al Sprint es del 50%) no ha trabajado en las tareas de Sprint.

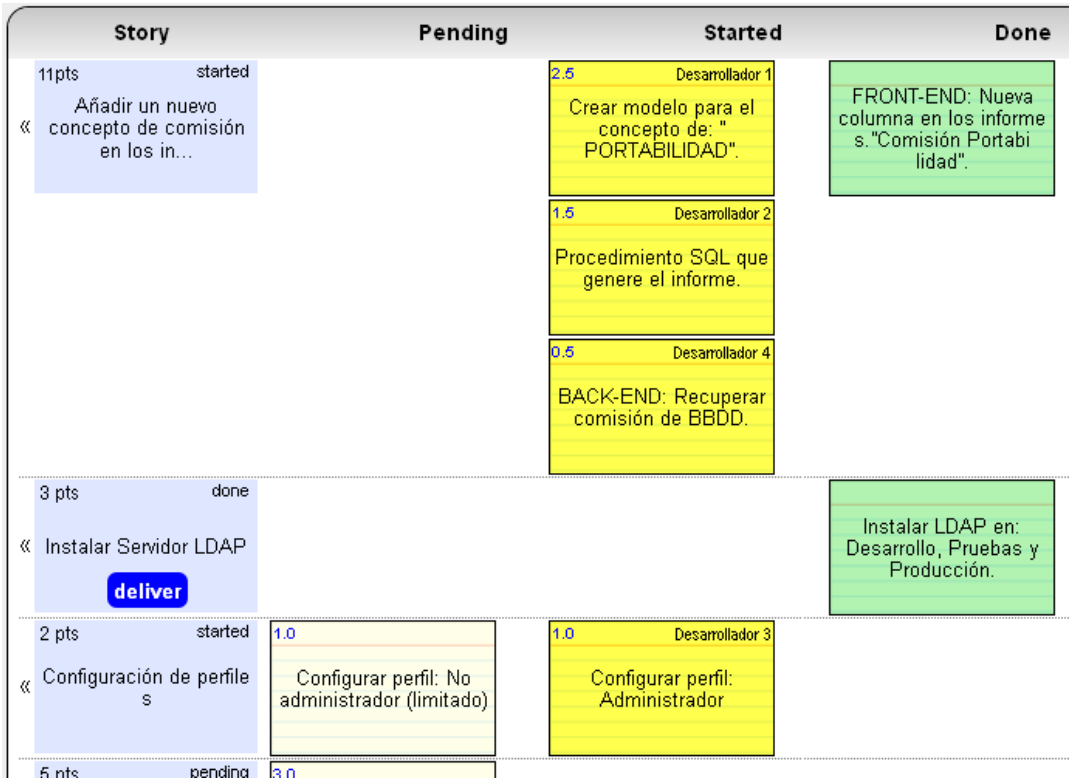


Ilustración 4-14: Caso Práctico - Tablón tercer día del Sprint

Estado actualizado de la gráfica

Se han quitado 3,5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 30,5 puntos de trabajo restantes.

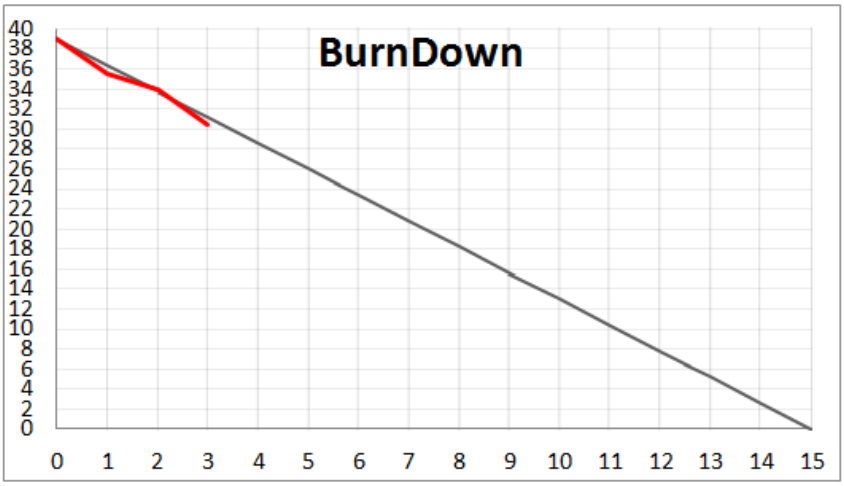


Ilustración 4-15: Caso Práctico - Burndown tercer día del Sprint

Día 4:

Desarrollador 1: Ha estado trabajando en la tarea: "Crear modelo para el concepto de PORTABILIDAD". Durante el día de hoy continuará con esta tarea. No ha encontrado ningún impedimento y estima que aún le quedan 1 puntos de trabajo.

Desarrollador 2: Ha estado trabajando en la tarea: "Procedimiento SQL que genere informe". Durante el día de hoy continuará con esta tarea. No hay encontrado ningún impedimento. Estima que aún le quedan 0.5 puntos de trabajo en esta tarea.

Desarrollador 3: Ha finalizado la tarea: "Configurar perfil: Administrador" y la coloca en la columna "**¡Hecho!**". Para continuar hoy se coge la tarea: "Configurar perfil: no Administrador (limitado)".

Desarrollador 4: Ha terminado la tarea: "BACK-END: Recuperar comisión de BBDD". Se coge una nueva tarea: "Perfil administrador: opción para simular distribuidor".

Desarrollador 5: El desarrollador 5 (su disponibilidad al Sprint es del 50%) no ha trabajado en las tareas de Sprint.

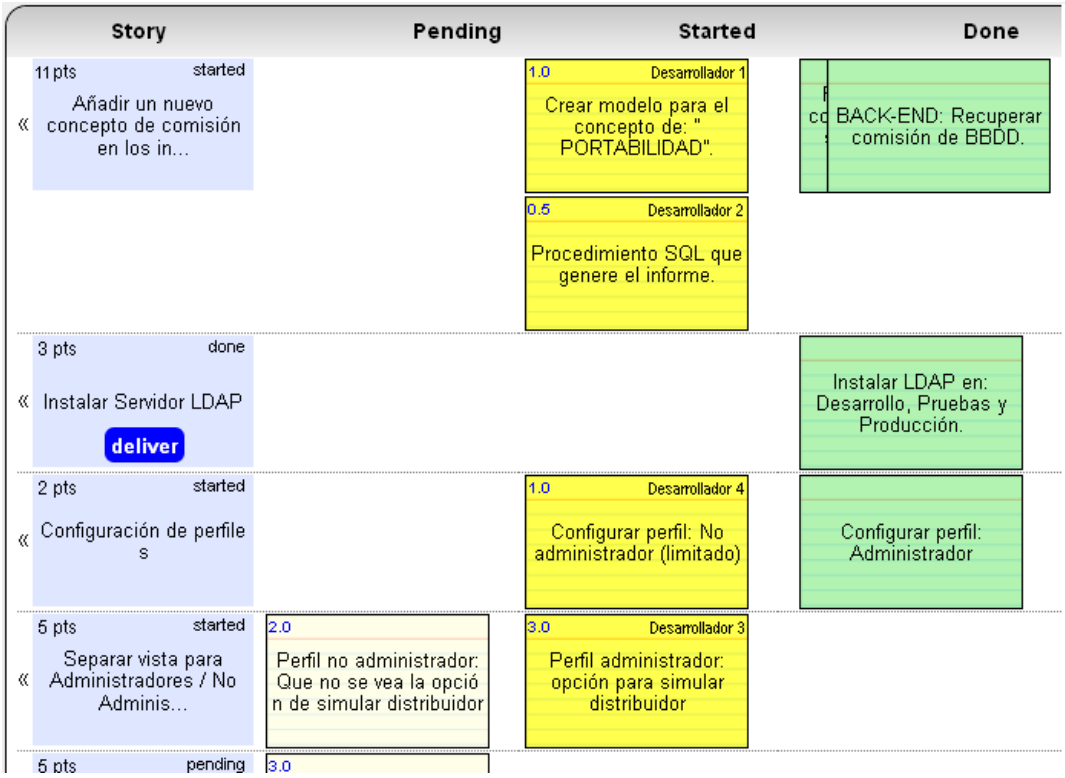


Ilustración 4-16: Caso Práctico - Tablón cuarto día del Sprint

Estado actualizado de la gráfica

Se han quitado 4 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 26.5 puntos de trabajo restantes.

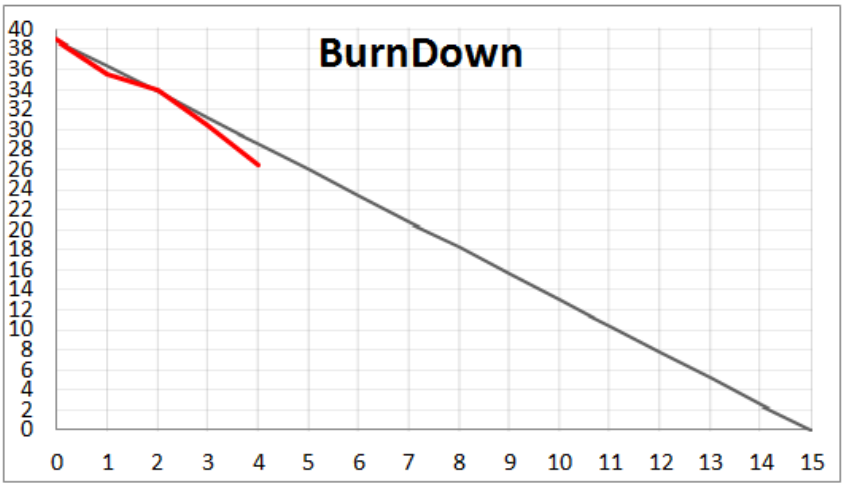


Ilustración 4-17: Caso Práctico - Burndown cuarto día del Sprint

Día 5:

Desarrollador 1: Fin de la tarea: "Crear modelo para el concepto de PORTABILIDAD", se mueve a la columna **"¡Hecho!"**. Se asigna la siguiente tarea disponible en el tablero: "Perfil no administrador: Que no se vea la opción de simular distribuidor".

Desarrollador 2: Fin de la tarea: "Procedimiento SQL que genere informe", se mueve a la columna **"¡Hecho!"**. Se asigna la siguiente tarea disponible en el tablero: "Incidencias / Subir fichero Excel".

Desarrollador 3: Ha trabajado en la tarea: "Perfil administrador: opción para simular distribuidor" reestima que aún le quedan 1.5 puntos de trabajo. Hoy se mantendrá trabajando en esta tarea.

Desarrollador 4: Ha finalizado la tarea: "Configurar perfil: no Administrador (limitado)" que pone en la columna "**¡Hecho!**". Coge la siguiente tarea disponible en el tablero: "Definir plantilla Excel".

Desarrollador 5: Coge la tarea: "Descargar plantilla Excel".

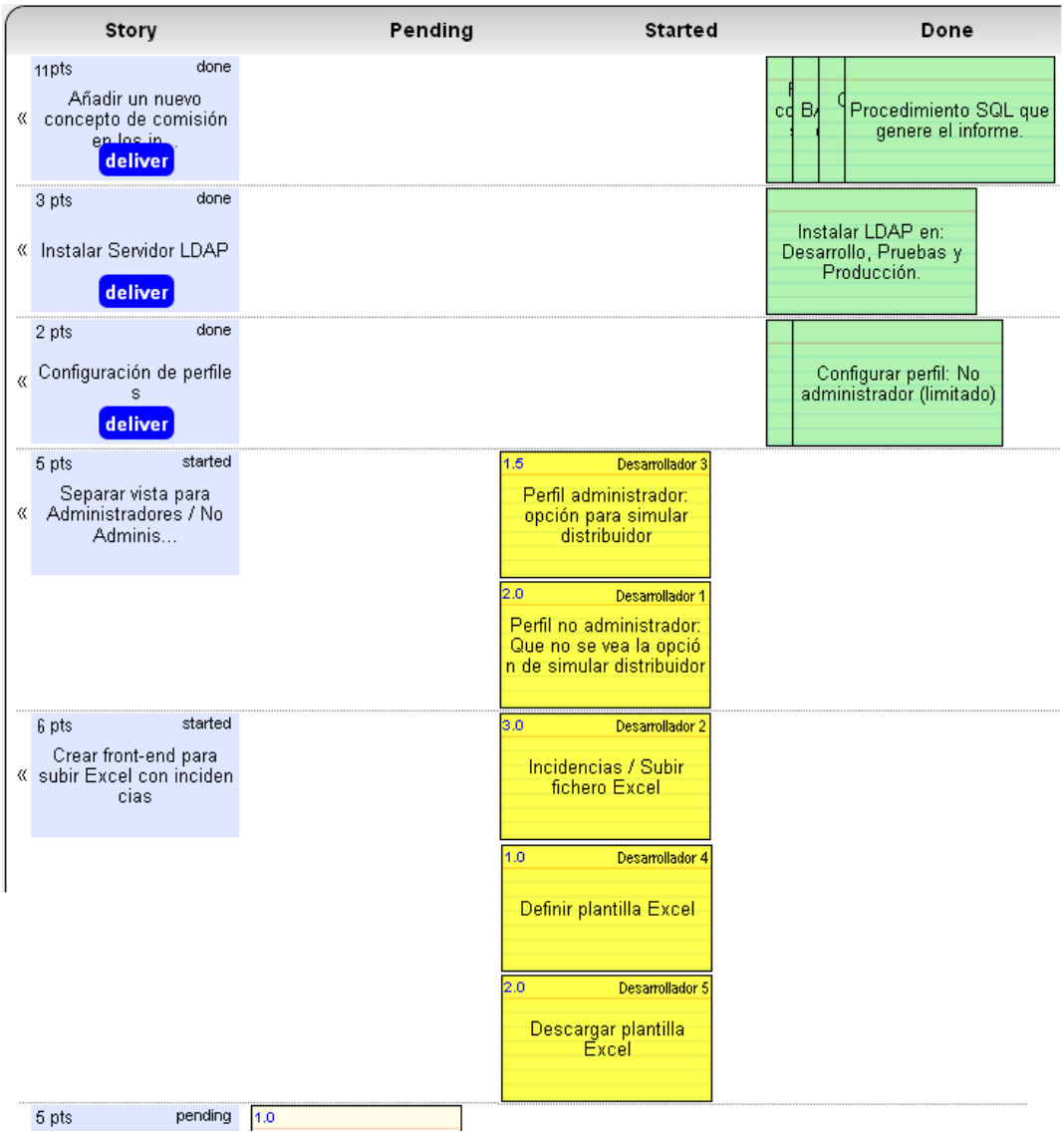


Ilustración 4-18: Caso Práctico - Tablón quinto día del Sprint

Estado actualizado de la gráfica

Se han quitado 4 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 22.5 puntos de trabajo restantes.

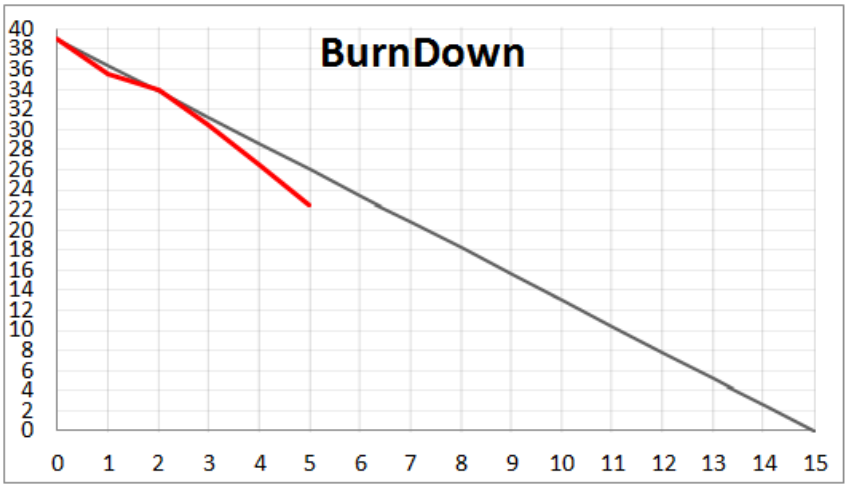


Ilustración 4-19: Caso Práctico - Burndown quinto día del Sprint

Día 6:

Desarrollador 1: Ha trabajado en la tarea: "Perfil no administrador: Que no se vea la opción de simular distribuidor". Reestima la tarea en 1 punto de trabajo.

Desarrollador 2: Ha trabajado en la tarea: "Incidencias / Subir fichero Excel". Reestima la tarea en 2.5 puntos de trabajo.

Desarrollador 3: Ha trabajado en la tarea: "Perfil administrador: opción para simular distribuidor" reestima que aún le quedan 0.5 puntos de trabajo. Hoy se mantendrá trabajando en esta tarea.

Desarrollador 4: Ha finalizado la tarea: "Definir plantilla Excel" que pone en la columna "**¡Hecho!**". Coge la siguiente tarea disponible en el tablero: "Tabla en BBDD".

Desarrollador 5: Ha trabajado en la tarea: "Descargar plantilla Excel". Reestima que aún le queda 1 punto de trabajo.

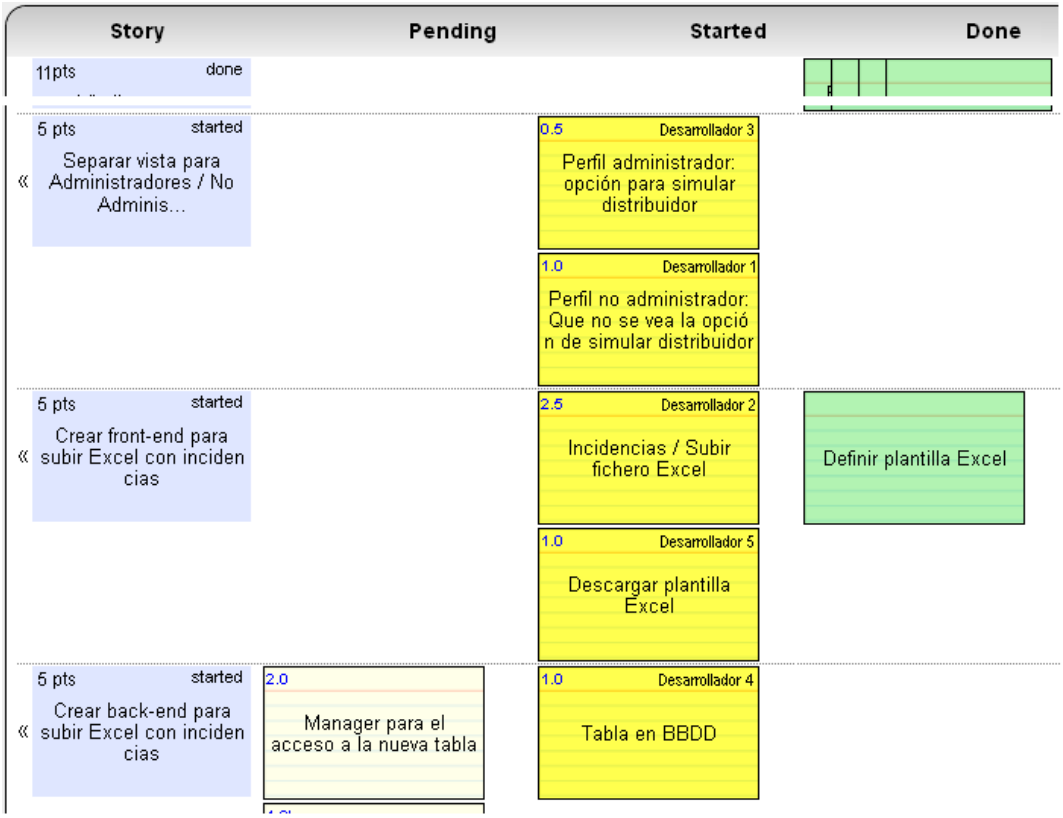


Ilustración 4-20: Caso Práctico - Tablón sexto día del Sprint

Estado actualizado de la gráfica

Se han quitado 4.5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 18 puntos de trabajo restantes.

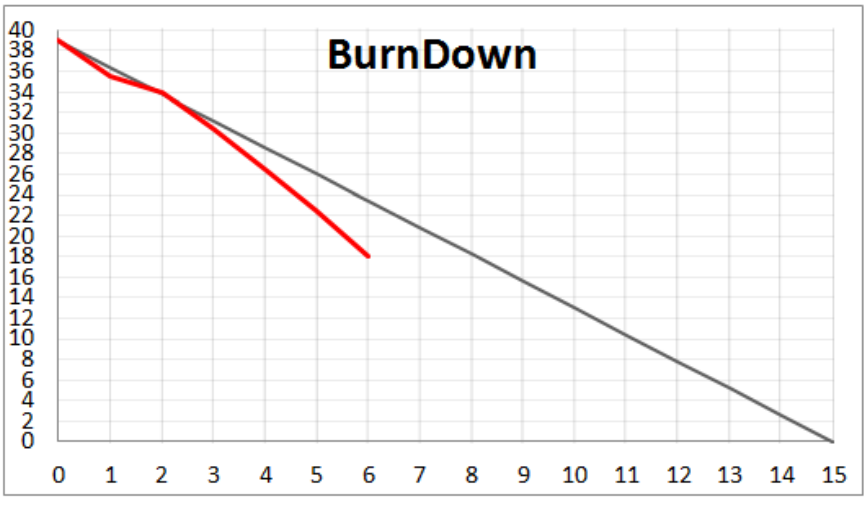


Ilustración 4-21: Caso Práctico - Burndown sexto día del Sprint

Día 7:

Desarrollador 1: Ha trabajado en la tarea: "Perfil no administrador: Que no se vea la opción de simular distribuidor". Mantiene la estimación en 1 punto de trabajo.

Desarrollador 2: Ha trabajado en la tarea: "Incidencias / Subir fichero Excel". Reestima la tarea en 1.5 de trabajo.

Desarrollador 3: Ha finalizado la tarea: "Perfil administrador: opción para simular distribuidor". Se coge la siguiente tarea disponible en el tablero: "Manager para el acceso a la nueva tabla".

Desarrollador 4: Ha trabajado en la tarea: "Tabla en BBDD". Reestima que aún le queda 1 punto de trabajo.

Desarrollador 5: Ha finalizado la tarea: "Descargar plantilla Excel". Se coge la siguiente tarea disponible en el tablero:"Recoger el archivo que sube el usuario y almacenarlo".

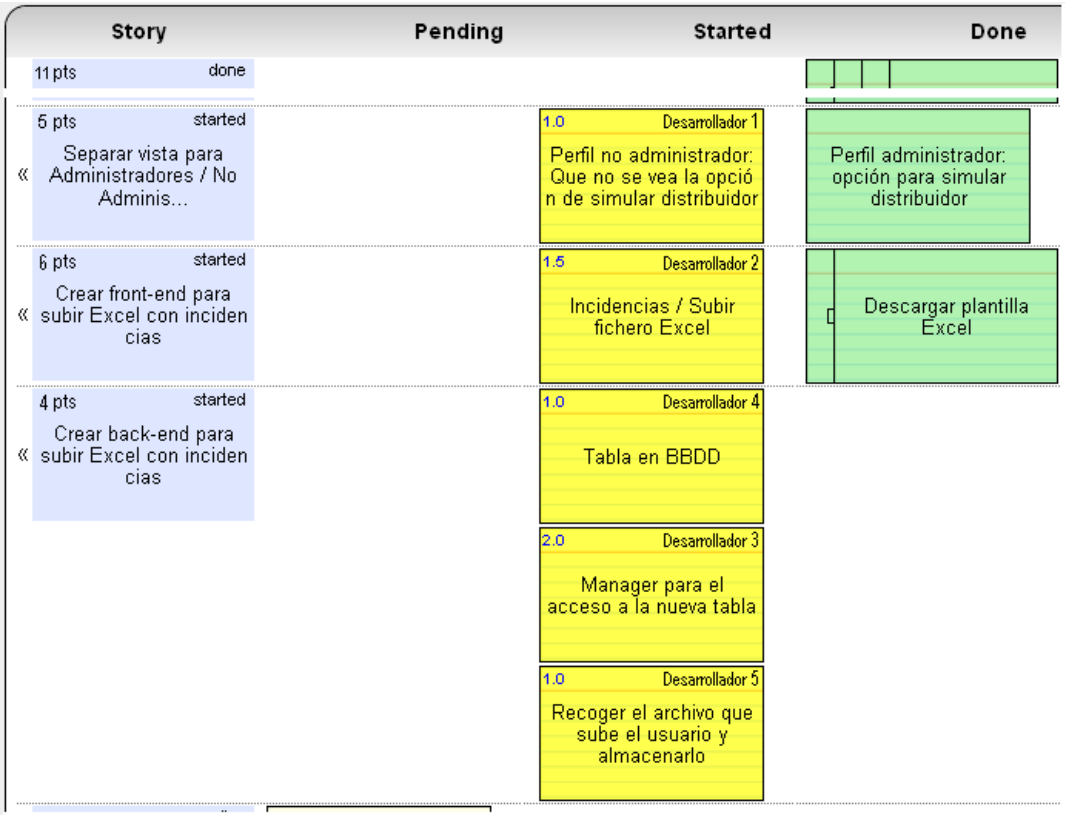


Ilustración 4-22: Caso Práctico - Tablón séptimo día del Sprint

Estado actualizado de la gráfica

Se han quitado 2.5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 15.5 puntos de trabajo restantes.

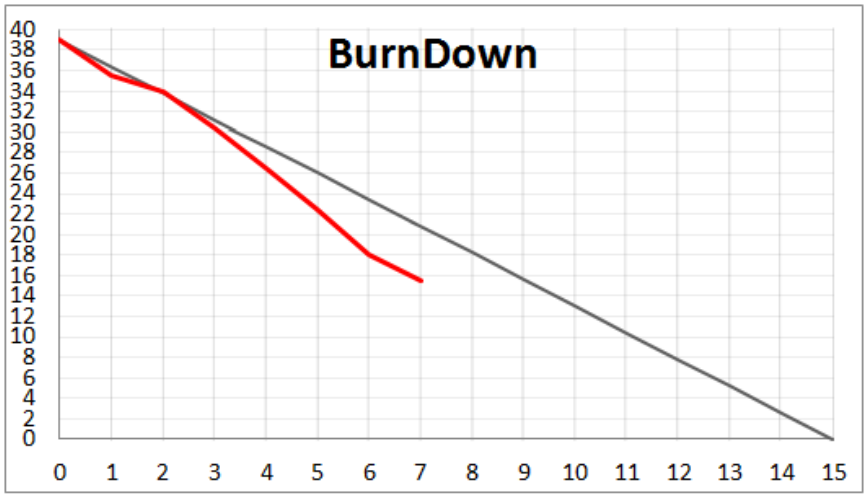


Ilustración 4-23: Práctico - Burndown séptimo día del Sprint

Día 8:

Desarrollador 1: Ha finalizado la tarea: "Perfil no administrador: Que no se vea la opción de simular distribuidor". Coge la siguiente tarea disponible en el tablero:"Script Perl para la creación de incidencias".

Desarrollador 2: Ha trabajado en la tarea: "Incidencias: Subir fichero Excel". Reestima la tarea en 0.5 puntos de trabajo.

Desarrollador 3: Ha trabajado en la tarea: "Manager para el acceso a la nueva tabla". Reestima la tarea en 1 punto de trabajo.

Desarrollador 4: Ha finalizado la tarea: "Tabla en BBDD". Coge la siguiente tarea disponible en el tablero:"Crear incidencias e insertarlas en BBDD".

Desarrollador 5: Ha finalizado la tarea en la que estaba trabajando: "Recoger el archivo que sube el usuario y almacenarlo". Coge la siguiente tarea disponible en el tablero:"Añadir validaciones al script".

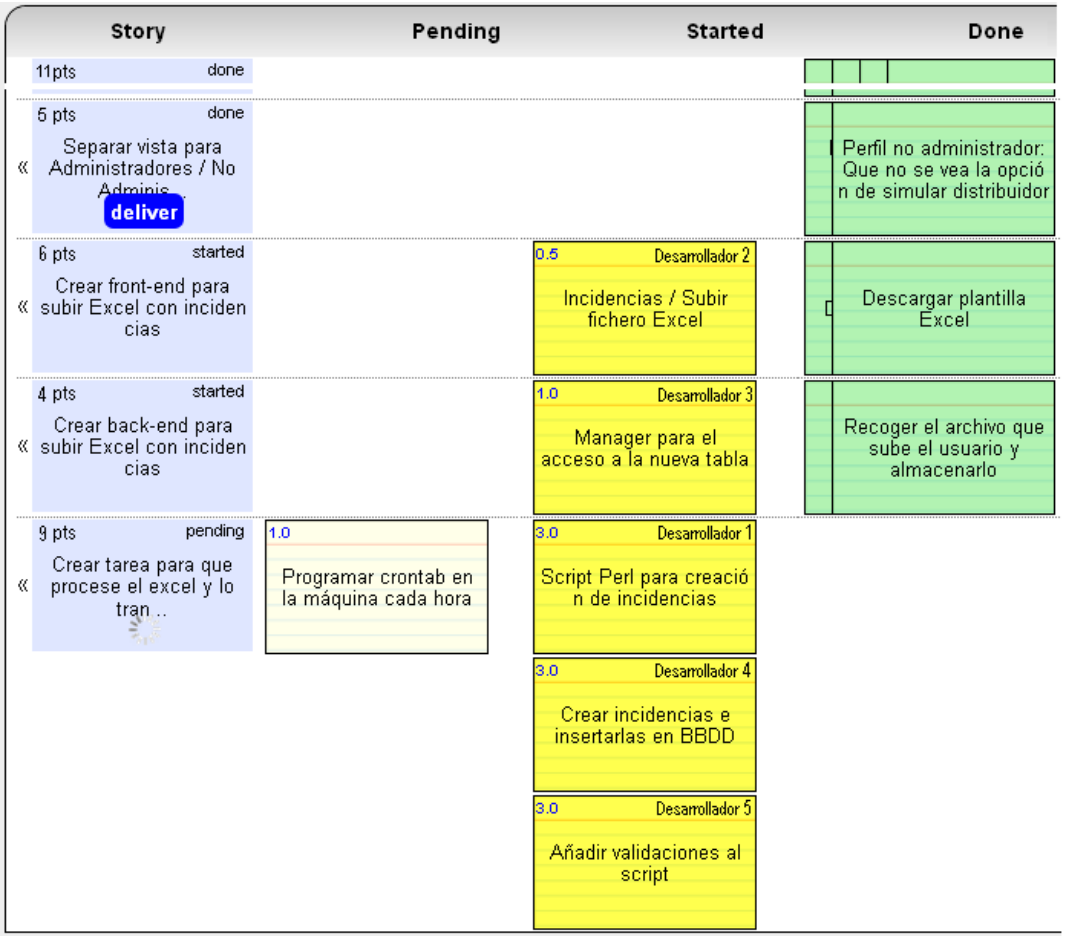


Ilustración 4-24: Caso Práctico - Tablón octavo día del Sprint

Estado actualizado de la gráfica

Se han quitado 5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 10.5 puntos de trabajo restantes.

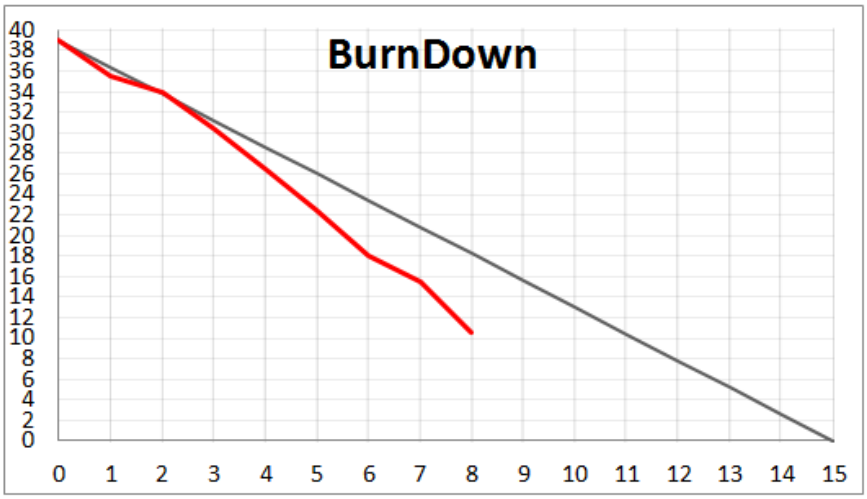


Ilustración 4-25: Caso Práctico - Burndown octavo día del Sprint

Día 9:

Desarrollador 1: Se mantiene trabajando en la tarea:"Script Perl para creación de incidencias". Mantiene la estimación de 3 puntos de trabajo.

Desarrollador 2: Se mantiene trabajando en la tarea:"Incidencias / Subir fichero Excel". Mantiene la estimación de 0.5 puntos de trabajo.

Desarrollador 3: Se mantiene trabajando en la tarea:"Manager para el acceso a la nueva tabla". Mantiene la estimación de 1 puntos de trabajo.

Desarrollador 4: Ha trabajo en la tarea: "Crear incidencias e insertarlas en BBDD". Como la tarea que tiene el Desarrollador 1 aún no está terminada, considera que hay dependencias entre ambas y aumenta en un punto el tiempo estimado para su tarea.

Desarrollador 5: Ha trabajo en la tarea: "Añadir validaciones al Script". Tarea también dependiente de la tarea del desarrollador 1, por tanto aumenta en medio punto la estimación para su tarea.

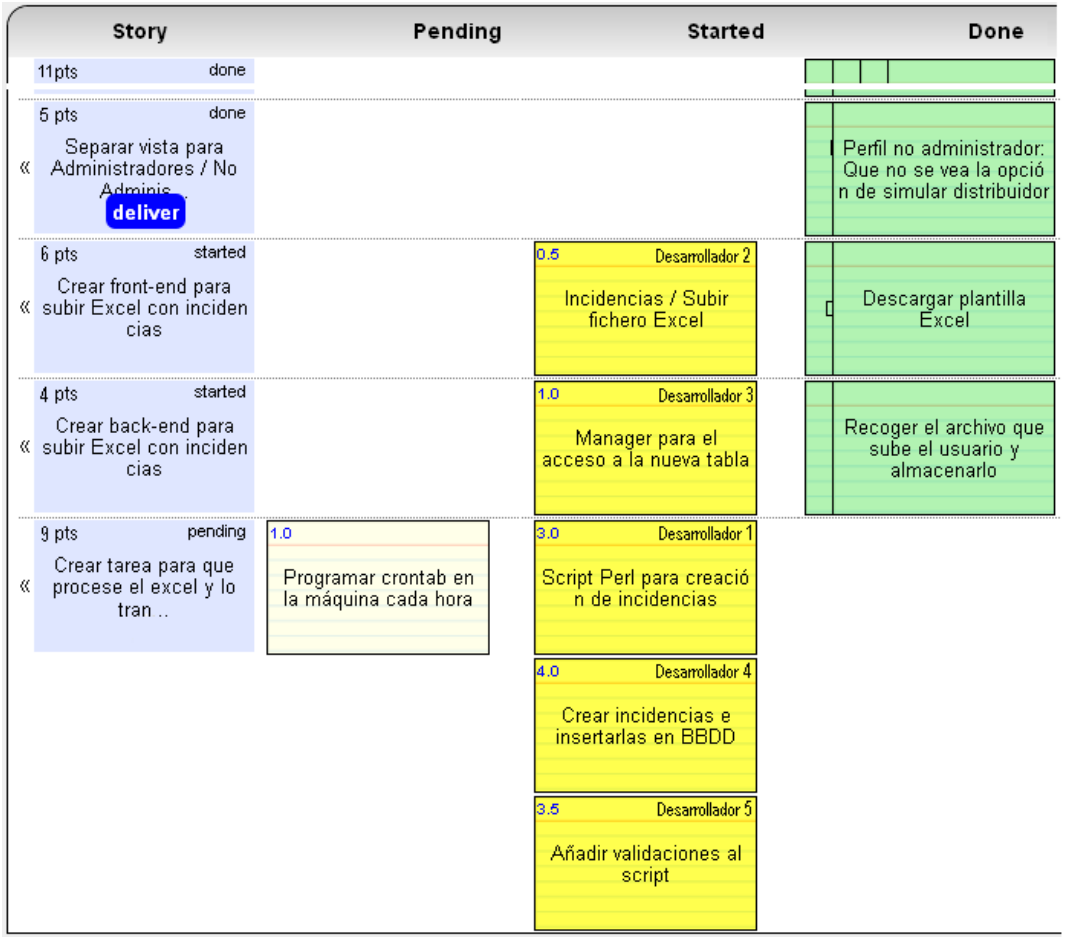


Ilustración 4-26: Caso Práctico - Tablón noveno día del Sprint

Estado actualizado de la gráfica

Se ha aumentado en 1.5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 12 puntos de trabajo restantes.

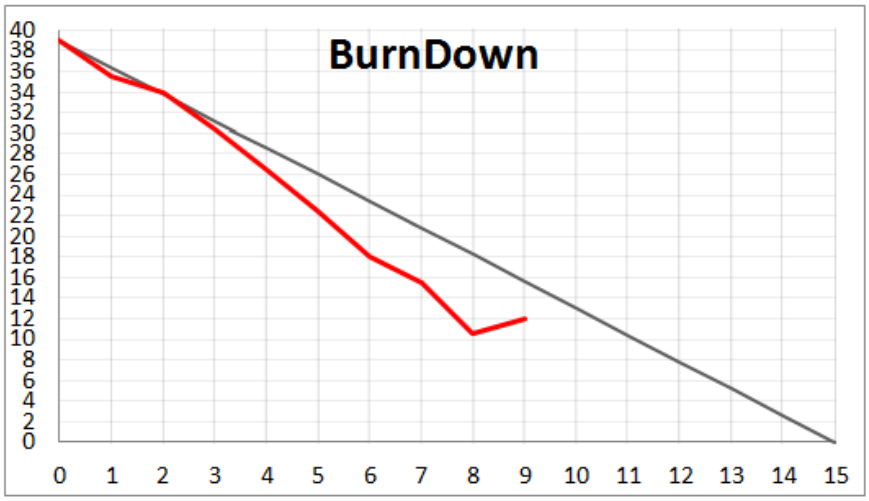


Ilustración 4-27: Caso Práctico - Burndown noveno día del Sprint

Día 10:

Día de reuniones ajenas al Sprint, ningún desarrollador avanza en el trabajo asignado, por tanto no se avanza en los puntos del Sprint.

Estado actualizado de la gráfica:

No se ha avanzado en el trabajo: Tenemos que marcar el punto de hoy de la gráfica en 12 puntos de trabajo restantes.

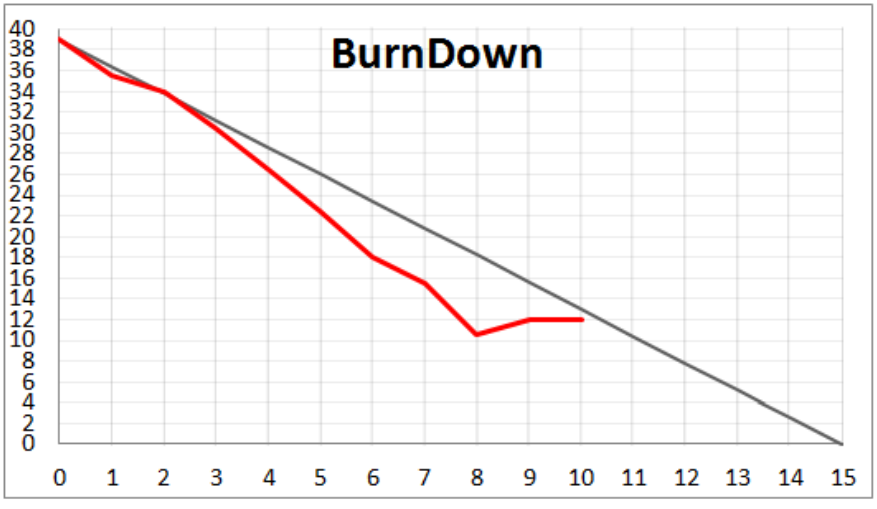


Ilustración 4-28: Práctico - Burndown décimo día del Sprint

Día 11:

Desarrollador 1: Ha avanzado en la tarea: "Script Perl para la creación de incidencias", la reestima en 2 puntos y continúa trabajando en ella.

Desarrollador 2: Ha finalizado la tarea: "Incidencias / Subir fichero Excel" y la coloca en la columna: "**¡Hecho!**". Coge la siguiente tarea disponible en el tablero:"".

Desarrollador 3: Continúa trabajando en la tarea: "Manager para el acceso a la nueva tabla" y mantiene la estimación en 1 punto de trabajo.

Desarrollador 4: Ha estado trabajando en la tarea: "Crear incidencias e insertarlas en BBDD", mantiene la estimación en 4 puntos de trabajo.

Desarrollador 5: Al estar bloqueado no ha trabajado en la tarea. Mantiene la estimación de 3.5 puntos de trabajo. Durante el día de hoy no trabajará en las tareas del Sprint.

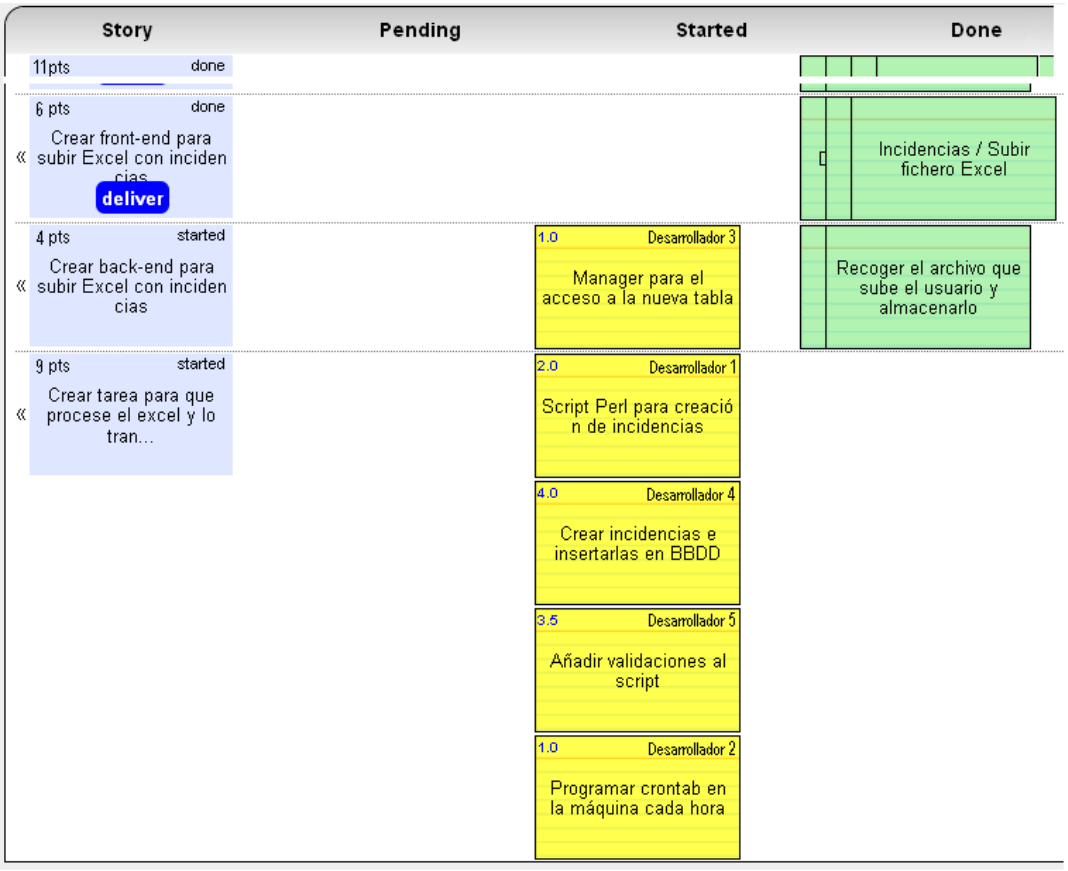


Ilustración 4-29: Caso Práctico - Tablón undécimo día del Sprint

Estado actualizado de la gráfica

Se han quitado 1.5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 10.5 puntos de trabajo restantes.

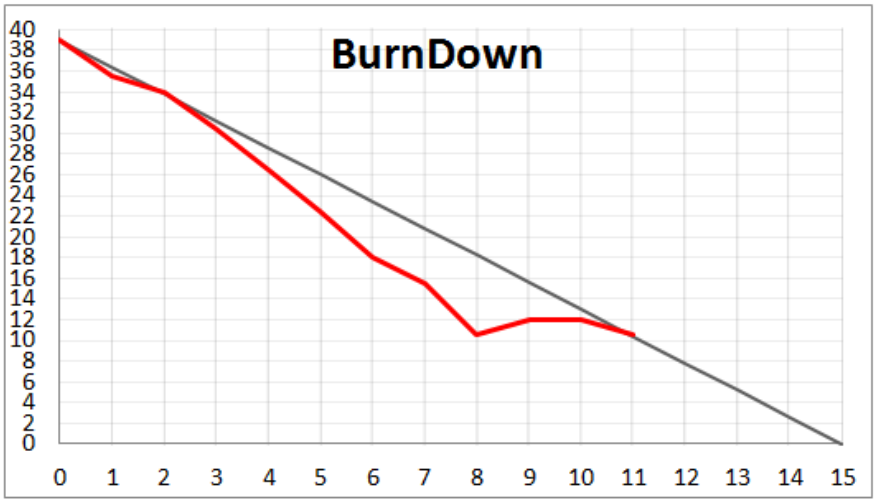


Ilustración 4-30: Caso Práctico - Burndown undécimo día del Sprint

Día 12:

Desarrollador 1: Ha finalizado la tarea: "Script Perl para la creación de incidencias", como no hay tareas nuevas se junta con otro compañero para finalizar tareas a las que todavía le quedan varios puntos por terminar. Se une al desarrollador 5.

Desarrollador 2: Se mantiene trabajando la tarea: "Programar contab". Y mantiene la estimación de 1 punto.

Desarrollador 3: Ha finalizado la tarea en la que estaba trabajando: "Manager para el acceso a la nueva tabla", por tanto, la deja ya en la columna de: "**¡Hecho!**". Como no quedan tareas nuevas en la que trabajar, se une al trabajo de algún compañero de trabajo al que aún le queden muchos puntos por hacer. Se une al desarrollador 4.

Desarrollador 4: Se mantiene trabajando en la tarea: "Crear incidencias e insertarlas en BBDD", ya se ha finalizado la tarea: "Script Perl para creación de incidencias" y ya no está bloqueado por tanto tendrá que recuperar el retraso que haya podido acumular en el bloqueo. El desarrollador 3 se une para trabajar en esta misma tarea.

Desarrollador 5: Se mantiene trabajando en la tarea: "Añadir validaciones al script", ya se ha finalizado la tarea: "Script Perl para creación de incidencias" y ya no está bloqueado por tanto tendrá que recuperar el retraso que haya podido acumular en el bloqueo. El desarrollador 1 se une para trabajar en esta misma tarea.

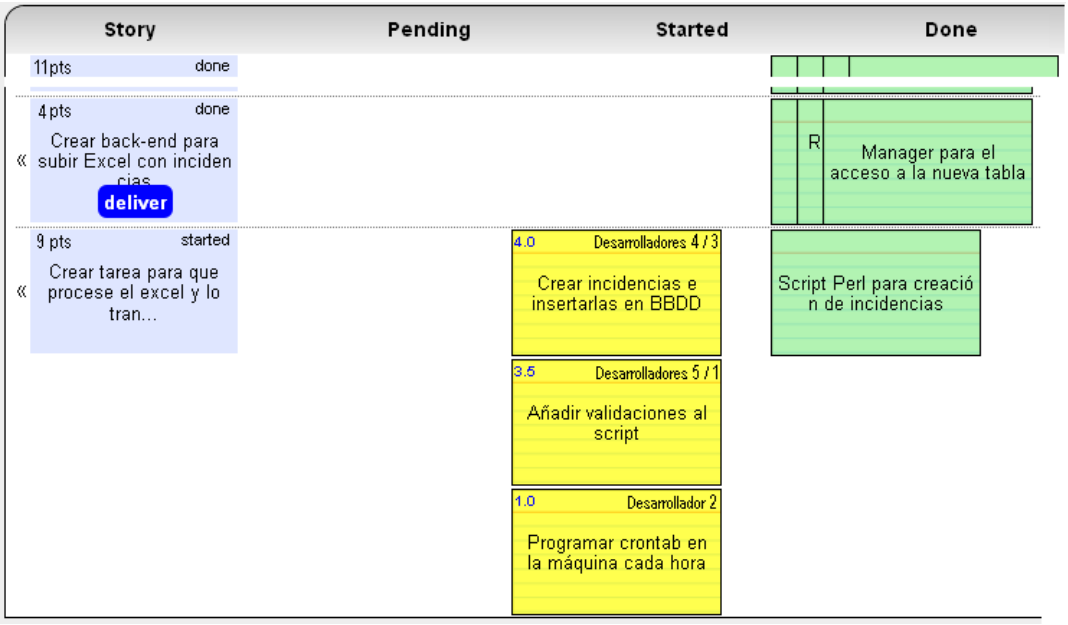


Ilustración 4-31: Caso Práctico - Tablón decimosegundo día del Sprint

Estado actualizado de la gráfica

Se han quitado 3 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 7.5 puntos de trabajo restantes.

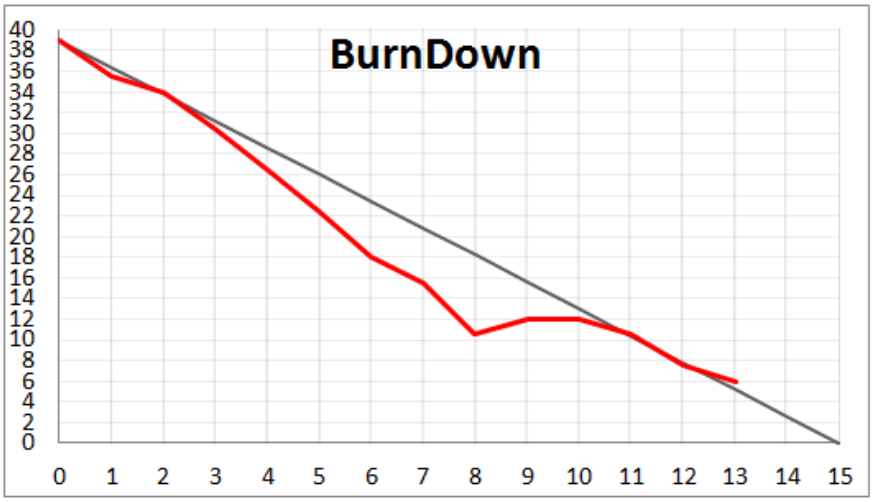


Ilustración 4-32: Caso Práctico - Burndown decimosegundo día del Sprint

Día 13:

Desarrollador 1 y desarrollador 5: Continúan trabajando en la tarea: "Crear incidencias e insertarlas en BBDD".

Desarrollador 2: Finaliza la tarea que estaba realizando: "Programar crontab". Comienza la preparación de la demo.

Desarrollador 3 y desarrollador 4: Continúan trabajando en la tarea: "Añadir validaciones al script".

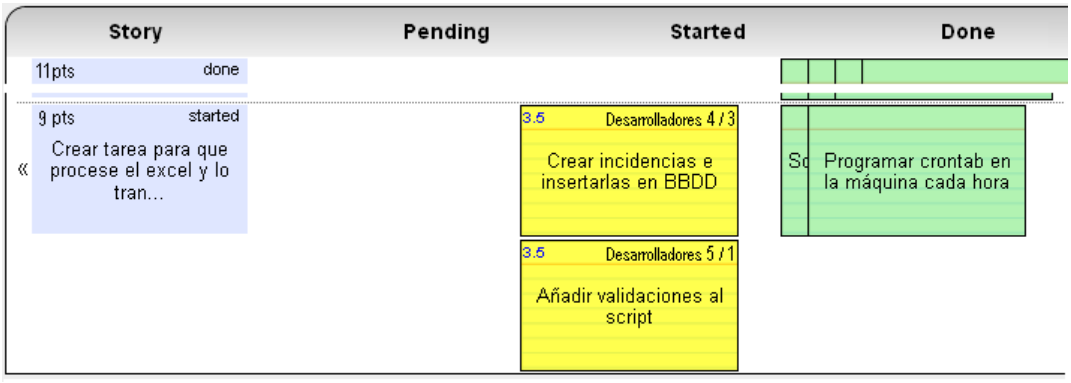


Ilustración 4-33: Caso Práctico - Tablón decimotercer día del Sprint

Estado actualizado de la gráfica

Se han quitado 1.5 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 6 puntos de trabajo restantes.

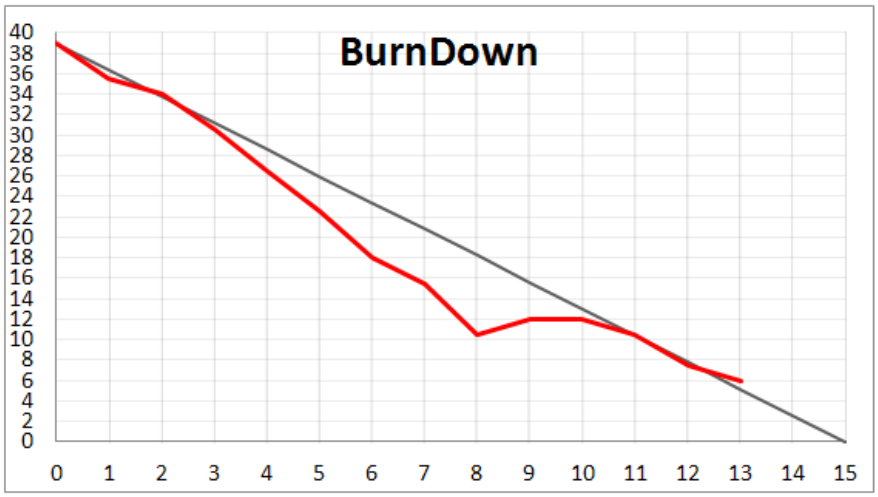


Ilustración 4-34: Caso Práctico - Burndown decimotercer día del Sprint

Día 14:

Desarrollador 1 y desarrollador 5: Continúan trabajando en la tarea: "Crear incidencias e insertarlas en BBDD", estiman 2.5 puntos de trabajo.

Desarrollador 2: Preparación de la demo.

Desarrollador 3 y desarrollador 4: Continúan trabajando en la tarea: "Añadir validaciones al script". Estiman 2.5 puntos de trabajo.

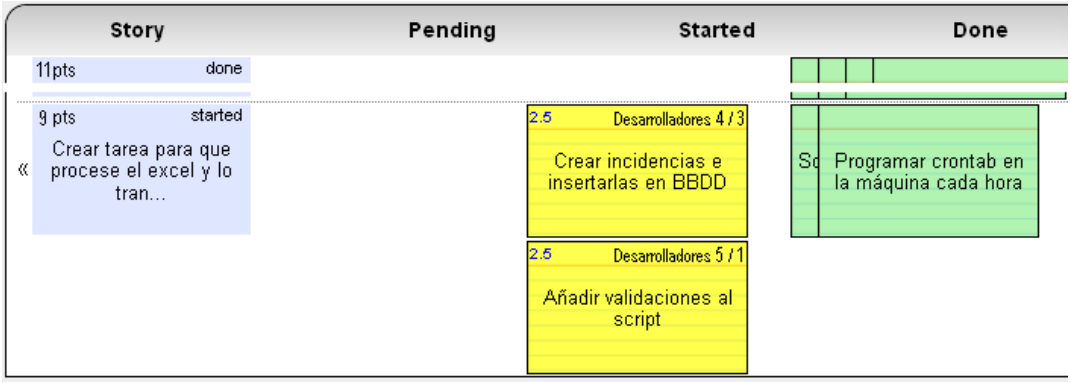


Ilustración 4-35: Caso Práctico - Tablón decimocuarto día del Sprint

Estado actualizado de la gráfica

Se han quitado 2 puntos de trabajo: Tenemos que marcar el punto de hoy de la gráfica en 4 puntos de trabajo restantes.

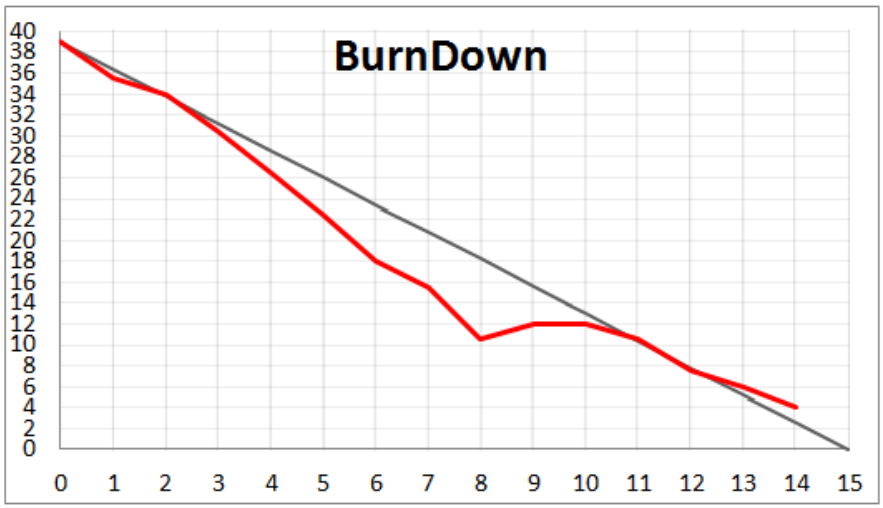


Ilustración 4-36: Caso Práctico - Burndown decimocuarto día del Sprint

Día 15:

Desarrollador 1 y desarrollador 5: Finalizan la tarea: "Crear incidencias e insertarlas en BBDD", se mueve a la columna "**¡Hecho!**". Se unen a la preparación de la demo.

Desarrollador 2: Preparación de la demo.

Desarrollador 3 y desarrollador 4: Han trabajado en la tarea: "Añadir validaciones al script", pero no han terminado, estiman que quedan 0.5 puntos de trabajo.

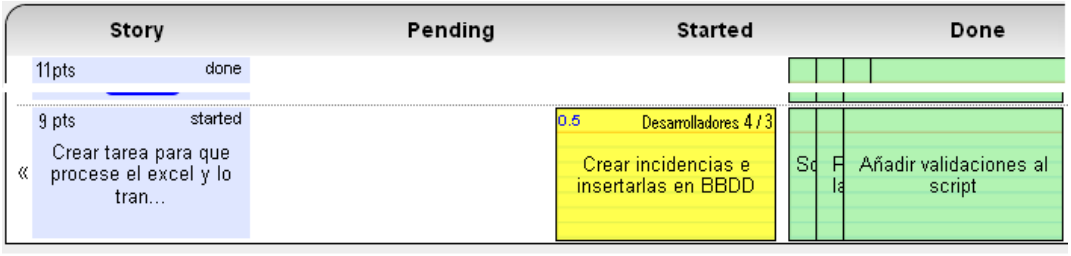


Ilustración 4-37: Caso Práctico - Tablón último día del Sprint

Estado final de la gráfica

Han faltado por completar medio punto de trabajo.

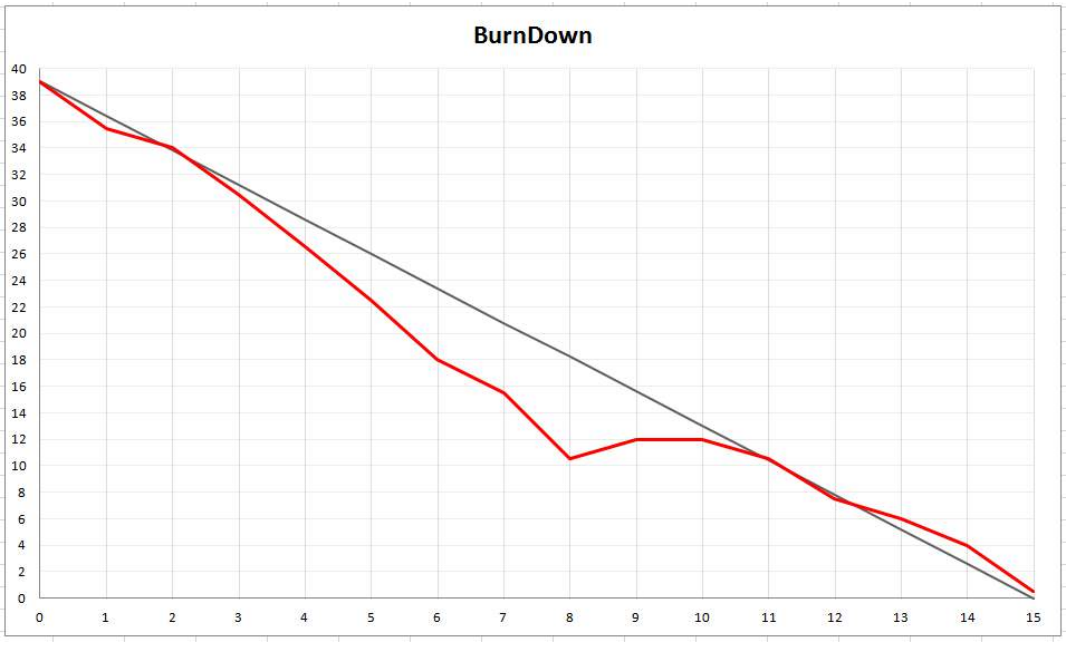


Ilustración 4-38: Caso Práctico - Burndown último día del Sprint

Demostración

Una vez preparado el guión para la reunión, se convoca al cliente a una reunión en la que ha visto las nuevas funcionalidades que se han añadido al aplicativo.

Reunión retrospectiva

Una vez finalizado el Sprint se realiza la reunión retrospectiva, el objetivo de la reunión es reflexionar para mejorar el proceso: como está trabajando el equipo junto, prácticas o herramientas que se están utilizando.

Esta reunión está restringida a los miembros del equipo.

Las conclusiones se encuentran resumidas a continuación:

Positivo	Mejorable	Soluciones	Ideas
Mejora tareas QA			Crear el guión de la demo como otro documento del sprint
Estimaciones cumplidas			Conocer infraestructura
Requisitos claros para el equipo	Mejorar la visión de los requisitos para el "cliente"	Dar más visibilidad de los requerimientos incluyendo a los usuarios finales de los mismos (no solo al owner como jefe)	Aportar/plantear posibles desarrollos que puedan interesar a negocio
Preparación Demo			
Guía de Tests para los usuarios	Intentar que los usuarios no se ciñan a la guía		
Mejorar comunicación (QA-Equipo de desarrollo)			
	Mejorar la toma de requisitos durante el Sprint para el siguiente Sprint	Tener en cuenta las fechas de SMT con las planificaciones de sprint	
	No se ha llegado a la fecha del Sprint	Planificar la demo para 2 días después del fin de desarrollo de sprint para permitir tiempo de preparación, resolución de posibles problemas de última hora	
	Entornos / servicios preparados	Invertir tiempo en repasar y montar si es necesario los entornos de desarrollo, QA, etc.	
	Retraso de la demo para mostrar toda la funcionalidad	Se debería haber hecho la demo con las funcionalidades desarrolladas	
		Resolver impedimentos en cuanto aparecen	

Tabla 4-2: Reunión retrospectiva

Capítulo 5: Planificación

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Contenido

Planificación

A la hora de desarrollar este Trabajo Fin de Grado se ha seguido la metodología Scrum. Se ha considerado oportuno trabajar con iteraciones de dos semanas de duración y se aplicará un factor de corrección de: 0,8

Backlog

ID	Imp	Proyecto	Nombre	Notas	Cómo probarlo
1	6	PFG - Guía comparativa de metodologías ágiles	Seleccionar metodologías ágiles a estudiar	Seleccionar una lista delimitada de metodologías sobre las que se va a desarrollar el PFG.	N/A
2	5	PFG - Guía comparativa de metodologías ágiles	Documentar cada una de las metodologías ágiles	Elaborar una documentación sobre las metodologías de estudio que permita conocer los aspectos más importantes de cada una de ellas.	N/A
3	4	PFG - Guía comparativa de metodologías ágiles	Establecer criterios de comparación	Crear una lista de criterios de comparación entre las diferentes metodologías de estudio.	N/A
4	3	PFG - Guía comparativa de metodologías ágiles	Comparativa de las metodologías ágiles	Desarrollar la comparación entre las metodologías seleccionadas.	Comprobar cuál es la recomendación de la Guía elaborada para diferentes escenarios de trabajo conocidos e integrados en las metodologías ágiles. Si la respuesta coincide con la metodología que se está útilizando la guía puede ser fiable.
5	2	PFG - Guía comparativa de metodologías ágiles	Caso práctico Scrum	Documentar una iteración de un caso práctico de Scrum.	N/A
6	1	PFG - Guía comparativa de metodologías ágiles	Maquetación de la memoria	Maquetación de la memoria del PFG según los estándares definidos por la EUISG.	Verificar que los estándares de los PFG se han aplicado a la memoria

Tabla 5-1: Backlog de la Guia comparativa de metodologías ágiles

Primera iteración

Backlog Item #1
Proyecto: PFG - Guía comparativa de metodologías ágiles

Seleccionar metodologías ágiles a estudiar

Notas

Seleccionar una lista delimitada de metodologías sobre las que se va a desarrollar el PFG.

Prioridad

4

Como probarlo

N/A

Estimación

2

Backlog Item #2
Proyecto: PFG - Guía comparativa de metodologías ágiles

Documentar teoría Scrum

Notas

Elaborar una documentación sobre Scrum que permita conocer los aspectos más importantes.

Prioridad

3

Como probarlo

N/A

Estimación

2

Backlog Item #3
Proyecto: PFG - Guía comparativa de metodologías ágiles

Documentar teoría XP

Notas

Elaborar una documentación sobre XP que permita conocer los aspectos más importantes.

Prioridad

2

Como probarlo

N/A

Estimación

2

Backlog Item #4
Proyecto: PFG - Guía comparativa de metodologías ágiles

Documentar teoría Kanban

Notas

Elaborar una documentación sobre Kanban que permita conocer los aspectos más importantes.

Prioridad

1

Como probarlo

N/A

Estimación

2

Si se trabaja con iteraciones de dos semanas, se dispone de diez puntos ideales de trabajo.

Se aplica el factor de corrección: 10 pto. Ideales * 0,8 = **8**

Este resultado quiere decir que en la primera iteración se van a asumir 8 puntos de trabajo.

Tomando las tareas del backlog por orden de prioridad, dentro de esta iteración se podrían desarrollar las tareas: "Seleccionar las metodologías ágiles a estudiar" y "Documentar cada una de las metodologías". Como la segunda no se podría completar en la primera iteración porque se pasaría de los puntos que disponemos, se ha decidido dividir aún más la historia de usuario. Como se ha decidido estudiar cuatro metodologías ágiles la segunda tarea se va a dividir en cuatro de la siguiente manera:

Historia de Usuario Original	División de la Historia de Usuario	Estimación
Documentar cada una de las metodologías ágiles	Documentar teoría Scrum	2
	Documentar teoría XP	2
	Documentar teoría Kanban	2
	Documentar teoría Scrumban	2
Total:		**8 ptos**

Tabla 5-2: División de la tarea Documentar cada metodología ágil en subtareas

De esta manera se podrán asumir en la primera iteración las siguientes historias de usuario:

Historia de Usuario	Estimación
Seleccionar metodologías ágiles a estudiar	2
Documentar teoría Scrum	2
Documentar teoría XP	2
Documentar teoría Kanban	2
Total:	**8 ptos**

Tabla 5-3: Historias de usuario de la primera iteración

Segunda iteración

Backlog Item #5
Proyecto: PFG - Guía comparativa de metodologías ágiles

Documentar teoría Scrumban

Notas
Elaborar una documentación sobre Scrumban que permita conocer los aspectos más importantes.

Prioridad
3

Como probarlo
N/A

Estimación
2

Backlog Item #6
Proyecto: PFG - Guía comparativa de metodologías ágiles

Establecer criterios de comparación

Notas
Crear una lista de criterios de comparación entre las diferentes metodologías de estudio.

Prioridad
2

Como probarlo
N/A

Estimación
4

Backlog Item #7
Proyecto: PFG - Guía comparativa de metodologías ágiles

Comparativa: Base teórica para orientación de la organización

Notas

Rasgos de una organización que sigue metologías tradiciones y rasgos de una organización ágil..

Prioridad

1

Como probarlo

Comprobar que hay suficientes detalles para distinguir con claridad si una organización es tradicional / ágil.

Estimación

2

Igual que en la primera iteración se dispone de diez puntos ideales de trabajo.

Se aplica el factor de corrección: 10 pto. Ideales * 0,8 = **8**

En la segunda iteración se asumen nuevamente 8 puntos de trabajo.

Siguiendo el orden de las tareas del backlog, dentro de esta segunda iteración se podría desarrollar la tarea: "Documentar teoría Scrumban", "Criterios de comparación" y "Comparativa de metodologías ágiles" estimada en once puntos ideales. Como no se pueden asumir más puntos de trabajo que los que se han calculado para la iteración, se divide la tarea más grande en tareas más pequeñas de la siguiente manera:

Historia de Usuario Original	**División de la Historia de Usuario**	**Estimación**
Comparativa de metodologías ágiles	Base teórica para orientación de la organización	2
	Base teórica para selección de la metodología ágil	2
	Elaboración del cuestionario	3
	Interpretación	3
	Ejemplo	1
Total:		**8 ptos**

Tabla 5-4: División de la tarea Comparativa de metodologías ágiles

De esta manera se podrán asumir en la primera iteración las siguientes historias de usuario:

Historia de Usuario	Estimación
Documentar teoría Scrumban	2
Criterios de comparación	4
Base teórica para orientación de la organización	2
Total:	**8 ptos**

Tabla 5-5: Historias de usuario de la segunda iteración

Tercera iteración

Backlog Item #8
Proyecto: PFG - Guía comparativa de metodologías ágiles

Comparativa: Base teórica para selección de la metodología ágil

Notas

Rasgos de cada una de las metodologías ágiles a comparar: Scrum, XP, Kanban, Scrumban.

Prioridad

3

Como probarlo

Comprobar que se han cubierto las cuatro metodologías. Comprobar que se ha elaborado una lista unificada de rasgos de las cuatros metodologías ágiles.

Estimación

2

Backlog Item #9
Proyecto: PFG - Guía comparativa de metodologías ágiles

Comparativa: Elaboración del cuestionario

Notas

Formularios.

Prioridad

2

Como probarlo

Comprobar que son claros, precisos, comprensibles.

Estimación

3

Backlog Item #10
Proyecto: PFG - Guía comparativa de metodologías ágiles

Comparativa: Interpretación

Notas

Interpretación del vaciado de los cuestionarios.

Prioridad

1

Como probarlo

Indicar si la interpretación del vaciado de los cuestionarios es clara y comprensible.

Estimación

3

Igual que en la primera iteración se dispone de diez puntos ideales de trabajo.

Se aplica el factor de corrección: 10 pto. Ideales * 0,8 = **8**

En la tercera iteración se asumen nuevamente 8 puntos de trabajo.

Siguiendo el orden de las tareas del backlog, dentro de esta tercera iteración se asumen las tareas que no se han podido incluir la iteración anterior: "Base teórica para selección de la metodología ágil", "Elaboración del cuestionario" y "Interpretación".

Historia de Usuario	Estimación
Base teórica para selección de la metodología ágil	2
Elaboración del cuestionario	3
Interpretación	3
Total:	**8 ptos**

Tabla 5-6: Historias de usuario de la tercera iteración

Cuarta iteración

Backlog Item #11
Proyecto: PFG - Guía comparativa de metodologías ágiles

Comparativa: Ejemplo

Notas

Cumplimentar formularios e interpretar los resultados obtenidos.

Prioridad

3

Como probarlo

N/A

Estimación

1

Backlog Item #12
Proyecto: PFG - Guía comparativa de metodologías ágiles

Caso práctico: Planificación y Tareas

Notas

Acta de reunión de la reunión de planificación.

Prioridad

2

Como probarlo

N/A

Estimación

4

Backlog Item #13
Proyecto: PFG - Guía comparativa de metodologías ágiles

Caso práctico: Desarrollo

Notas

Documentar el día a día del progreso en el desarrollo de las nuevas funcionalidades.

Prioridad

1

Como probarlo

N/A

Estimación

3

Se dispone de diez puntos ideales de trabajo.

Se aplica el factor de corrección: 10 pto. Ideales * 0,8 = **8** por tanto, se asumen nuevamente 8 puntos de trabajo.

Siguiendo el orden de las tareas del backlog, dentro de esta cuarta iteración se asumen las siguientes tareas: "Ejemplo", "Caso Práctico Scrum". Una vez más la tarea completa del caso práctico no se puede asumir completa en una iteración, de manera que se divide en tareas más pequeñas de la siguiente manera:

Historia de Usuario Original	División de la Historia de Usuario	Estimación
Caso Práctico Scrum	Caso Práctico: Planificación y tareas	4
	Caso Práctico: Desarrollo	3
	Caso Práctico: Demostración	2
	Caso Práctico: Retrospectiva	1
Total:		**10 ptos**

Tabla 5-7: División de la tarea Caso Práctico Scrum en subtareas

De esta manera se podrán asumir las siguientes historias de usuario:

Historia de Usuario	Estimación
Ejemplo	1
Caso Práctico: Planificación y tareas	4
Caso Práctico: Desarrollo	3
Total:	**8 ptos**

Tabla 5-8: Historias de usuario de la cuarta iteración

Quinta iteración

Backlog Item #14
Proyecto: PFG - Guía comparativa de metodologías ágiles

Caso práctico: Demostración

Notas	Prioridad
Preparación de la demostración de las nuevas funcionalidades al cliente.	3

Como probarlo	Estimación
Comprobar que en el guión de la demostración aparecen todas las nuevas funcionalidades.	2

Backlog Item #15
Proyecto: PFG - Guía comparativa de metodologías ágiles

Caso práctico: Retrospectiva

Notas	Prioridad
Reunión retrospectiva del equipo de Scrum.	2

Como probarlo	Estimación
N/A	1

Backlog Item #16
Proyecto: PFG - Guía comparativa de metodologías ágiles

Maquetación de la memoria

Notas	Prioridad
Maquetación de la memoria del PFG según los estándares definidos por la EUISG.	1

Como probarlo	Estimación
Verificar que los estándares de los PFG se han aplicado a la memoria	3

Se dispone de diez puntos ideales de trabajo.

Se aplica el factor de corrección: 10 pto. Ideales * 0,8 = **8** por tanto, se asumen nuevamente 8 puntos de trabajo.

Siguiendo el orden de las tareas del backlog, dentro de esta última iteración todas las tareas restantes: "Caso Práctico: Demostración", "Caso Práctico: Retrospectiva" y "Maquetación de la memoria" y aún sobran dos puntos de trabajo.

Se asumen las siguientes historias de usuario:

Historia de Usuario	**Estimación**
Caso Práctico: Demostración	2
Caso Práctico: Retrospectiva	1
Maquetación de la memoria	3
Total:	**6 ptos**

Tabla 5-9: Historias de usuario de la quinta iteración

Diagrama de Gantt

Diagrama de Gantt del proyecto completo. Se reflejan las cinco iteraciones de Scrum que ha tomado la ejecución del TFC: "Guía Comparativa de Metodologías Ágiles".

	Nombre	Duración	Inicio	Terminado
1	**Guía Comparativa Metodologías Ágiles**	**48 days**	**4/06/12 8:00**	**8/08/12 17:00**
2	**Primera Iteración**	**10 days**	**4/06/12 8:00**	**15/06/12 17:00**
3	Seleccionar metodologías ágiles a estudiar	2 days	4/06/12 8:00	5/06/12 17:00
4	Documentar teoría Scrum	2 days	6/06/12 8:00	7/06/12 17:00
5	Documentar teoría XP	2 days	8/06/12 8:00	11/06/12 17:00
6	Documentar teoría Kanban	2 days	12/06/12 8:00	13/06/12 17:00
7	Factor de corrección	2 days	14/06/12 8:00	15/06/12 17:00
8	Seguimiento del proyecto	10 days	4/06/12 8:00	15/06/12 17:00
9	**Segunda Iteración**	**10 days**	**18/06/12 8:00**	**29/06/12 17:00**
10	Documentar teoría Scrumban	2 days	18/06/12 8:00	19/06/12 17:00
11	Establecer criterios de comparación	4 days	20/06/12 8:00	25/06/12 17:00
12	Comparativa: Base teórica para orientación de la organización	2 days	26/06/12 8:00	27/06/12 17:00
13	Factor de corrección	2 days	28/06/12 8:00	29/06/12 17:00
14	Seguimiento del proyecto	10 days	18/06/12 8:00	29/06/12 17:00
15	**Tercera Iteración**	**10 days**	**2/07/12 8:00**	**13/07/12 17:00**
16	Comparativa: Base teórica para selección de la metodología ágil	2 days	2/07/12 8:00	3/07/12 17:00
17	Comparativa: Elaboración del cuestionario	3 days	4/07/12 8:00	6/07/12 17:00
18	Comparativa: Interpretación	3 days	9/07/12 8:00	11/07/12 17:00
19	Factor de corrección	2 days	12/07/12 8:00	13/07/12 17:00
20	Seguimiento del proyecto	10 days	2/07/12 8:00	13/07/12 17:00
21	**Cuarta Iteración**	**10 days**	**16/07/12 8:00**	**27/07/12 17:00**
22	Comparativa: Ejemplo	1 day	16/07/12 8:00	16/07/12 17:00
23	Caso práctico: Planificación y Tareas	4 days	17/07/12 8:00	20/07/12 17:00
24	Caso práctico: Desarrollo	3 days	23/07/12 8:00	25/07/12 17:00
25	Factor de corrección	2 days	26/07/12 8:00	27/07/12 17:00
26	Seguimiento del proyecto	10 days	16/07/12 8:00	27/07/12 17:00
27	**Quinta Iteración**	**8 days**	**30/07/12 8:00**	**8/08/12 17:00**
28	Caso práctico: Demostración	2 days	30/07/12 8:00	31/07/12 17:00
29	Caso práctico: Retrospectiva	1 day	1/08/12 8:00	1/08/12 17:00
30	Maquetación de la memoria	3 days	2/08/12 8:00	6/08/12 17:00
31	Factor de corrección	2 days	7/08/12 8:00	8/08/12 17:00
32	Seguimiento del proyecto	8 days	30/07/12 8:00	8/08/12 17:00

Ilustración 5-1: Calendario del TFG

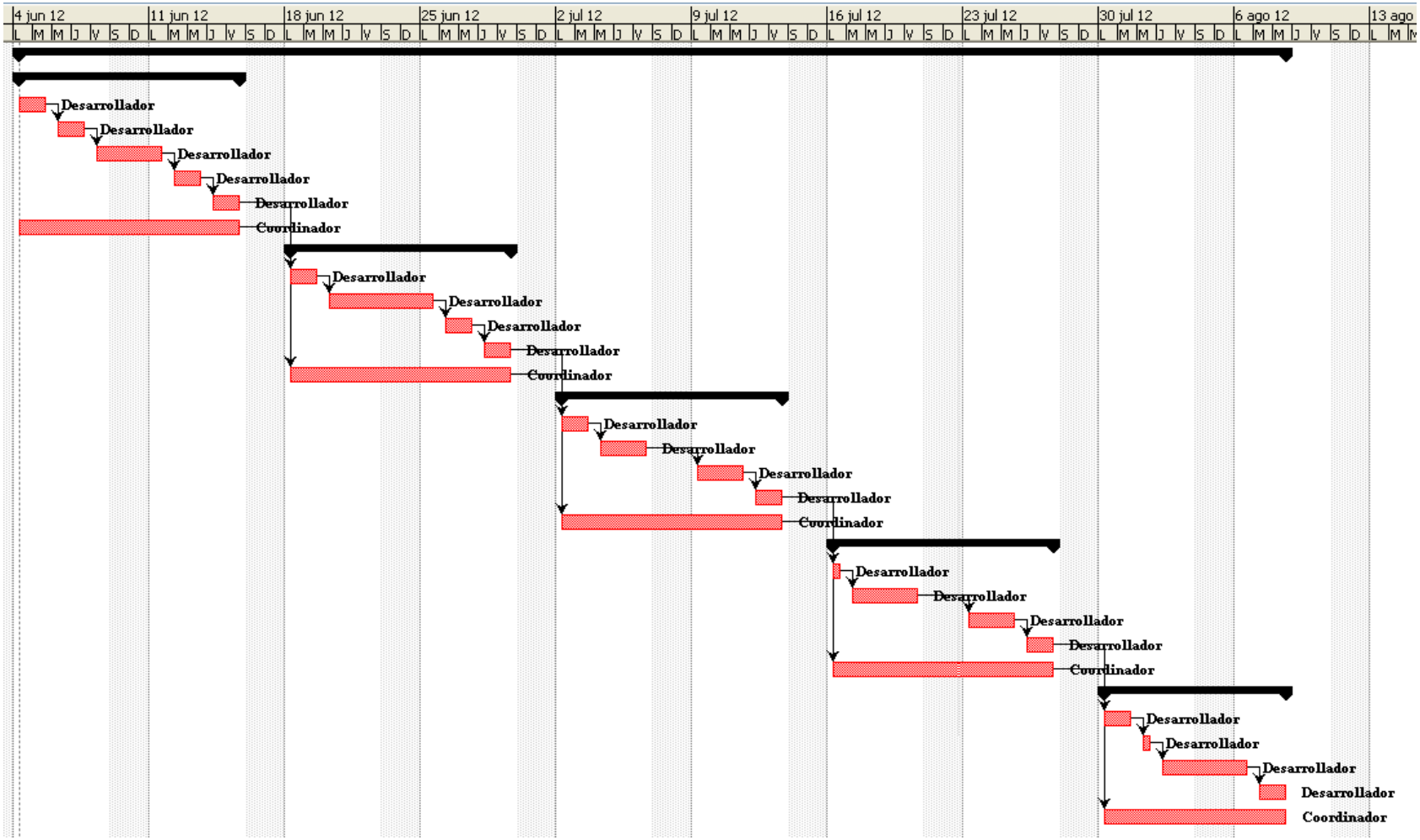


Ilustración 5-2: Diagrama de Gantt del TFG

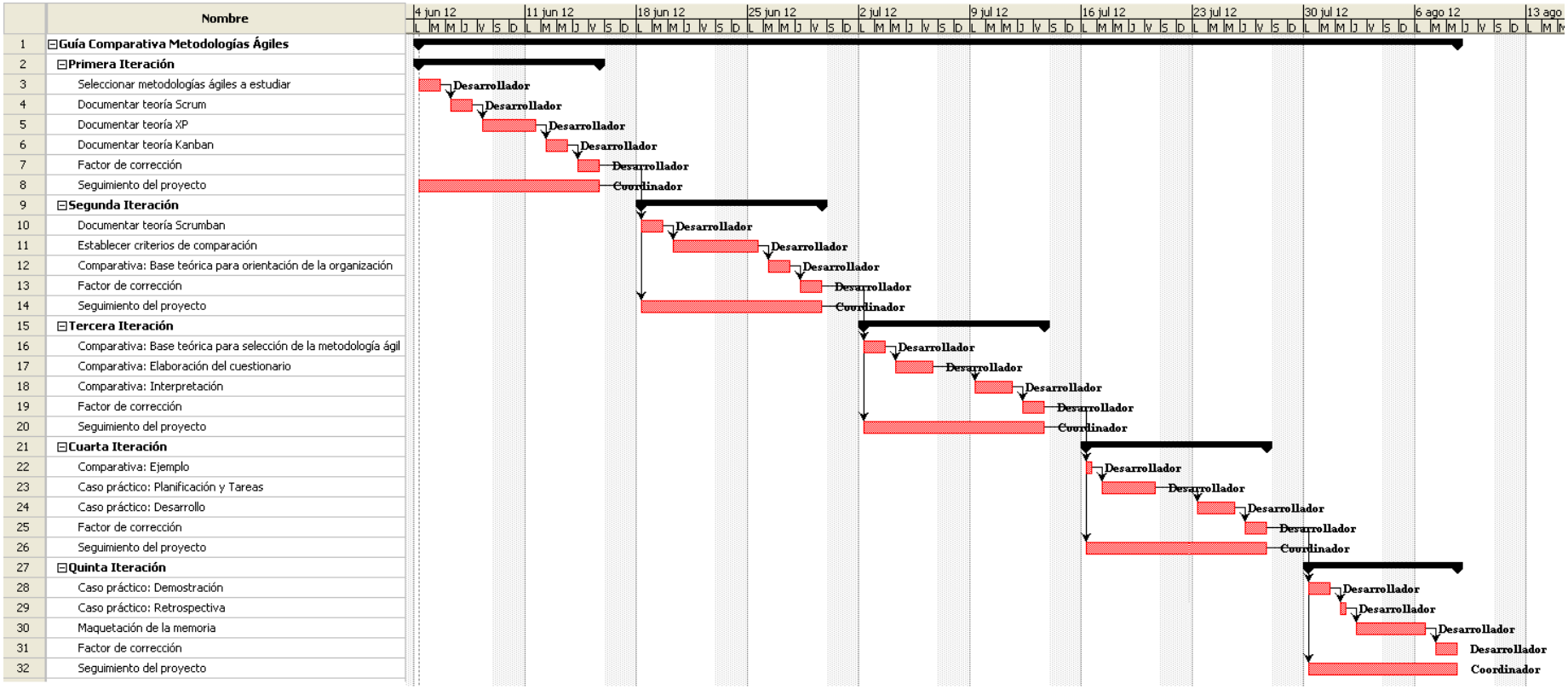


Ilustración 5-3: Calendario y diagrama de Gantt del TFG

Capítulo 6: Valoración económica

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Contenido

Valoración económica

Detalle de los costes del proyecto. Éstos se basan en dos aspectos: los recursos humanos y los recursos materiales.

Recursos Humanos

En el proyecto serán necesarios un coordinador de proyecto y un desarrollador / investigador.

Estimación de la duración de las actividades

Las horas de trabajo del coordinador será toda la duración del proyecto. Las del desarrollador / investigador corresponden a las tareas iniciales de análisis más a la definición y elaboración del proyecto y al testeo.

Al ser una única persona la que asume los diferentes roles las tareas se desarrollaran secuencialmente y por tanto, la dedicación del desarrollador / investigador será toda la duración del proyecto.

Costes unitarios

Tabla precio / hora será la siguiente para cada miembro del equipo

Proyecto	Persona Equipo	Precio Hora
PFG - Guía comparativa de metodologías ágiles	Coordinador proyecto	40 € / hora
PFG - Guía comparativa de metodologías ágiles	desarrollador	15 € / hora

Tabla 6-1: Precio / hora para cada miembro del equipo

Equipo y tiempo en el proyecto

Resumen de horas dedicadas al proyecto por cada miembro del equipo extraída de la herramienta OpenProj:

	Nombre	Trabajo
1	Desarrollador	384 horas
	Factor de corrección	*16 horas*
	Maquetación de la memoria	*24 horas*
	Factor de corrección	*16 horas*
	Documentar teoría Scrum	*16 horas*
	Documentar teoría XP	*16 horas*
	Caso práctico: Planificación y Tareas	*32 horas*
	Establecer criterios de comparación	*32 horas*
	Caso práctico: Demostración	*16 horas*
	Comparativa: Base teórica para orientación de la organización	*16 horas*
	Caso práctico: Retrospectiva	*8 horas*
	Factor de corrección	*16 horas*
	Comparativa: Interpretación	*24 horas*
	Factor de corrección	*16 horas*
	Documentar teoría Scrumban	*16 horas*
	Factor de corrección	*16 horas*
	Comparativa: Ejemplo	*8 horas*
	Comparativa: Elaboración del cuestionario	*24 horas*
	Caso práctico: Desarrollo	*24 horas*
	Documentar teoría Kanban	*16 horas*
	Seleccionar metodologías ágiles a estudiar	*16 horas*
	Comparativa: Base teórica para selección de la metodología ágil	*16 horas*
2	Coordinador	96 horas
	Seguimiento del proyecto	*16 horas*
	Seguimiento del proyecto	*20 horas*
	Seguimiento del proyecto	*20 horas*
	Seguimiento del proyecto	*20 horas*
	Seguimiento del proyecto	*20 horas*

Ilustración 6-1: Estimación y tiempo en el proyecto

Diagrama de Gantt agrupado por el nombre del recurso para ver cómo están distribuidas las horas de trabajo del coordinador y del desarrollador a lo largo del proyecto.

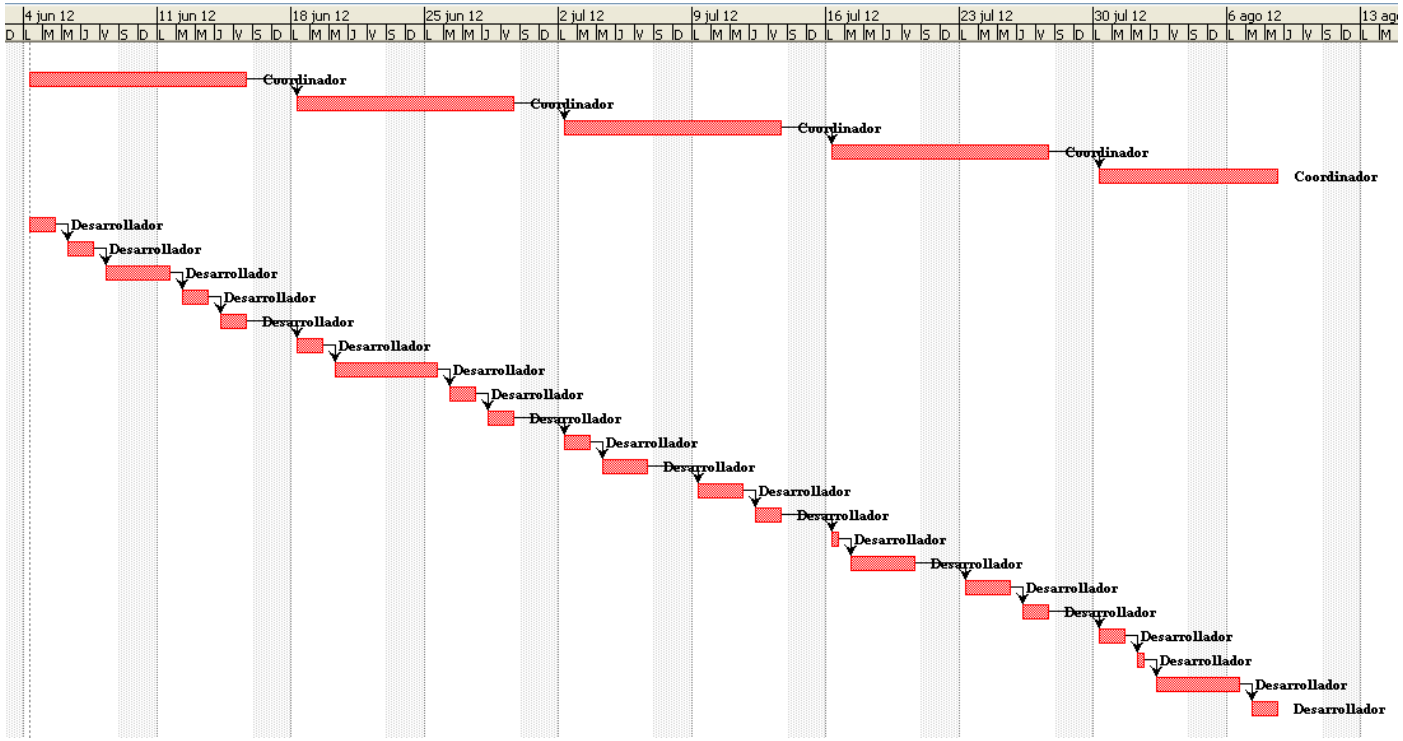


Ilustración 6-2: Gantt agrupado por recurso

Estimación de horas del proyecto

Se muestra la estimación en horas del proyecto, se ve como el coordinador del proyecto está durante todo el proyecto y el desarrollador va realizando las diferentes tareas de análisis, desarrollo y testeo a lo largo de todo el proyecto. Las tareas del desarrollador no se pueden solapar en ningún momento, ya que el proyecto sólo tiene un desarrollador asignado. En ningún caso el desarrollador está desocupado.

Mes	Junio	Julio	Agosto
Coordinador	40 h	44 h	12 h
Desarrollador	160 h	176 h	48 h
Total			**480 h**

Tabla 6-2: Horas del proyecto

Estimación económica del proyecto

A continuación la parte económica en euros teniendo en cuentas las horas del apartado anterior multiplicadas por el precio de cada recurso establecida en la tabla: "***Precio / hora para cada miembro del equipo***".

Coste mensual del proyecto

Mes	Junio	Julio	Agosto
Coordinador (40 €/h)	40 h	44 h	12 h
	1.600 €	1.760 €	480 €
Desarrollador (15 €/h)	160 h	176 h	48 h
	2.400 €	2.640 €	720 €
Total			**9600 €**

Tabla 6-3: Coste mensual del proyecto

Coste por Iteración y coste total extraídos de la herramienta OpenProj:

Primera iteración

Ilustración 6-3: Costo primera iteración

Segunda iteración

Ilustración 6-4: Costo segunda iteración

Tercera iteración

Ilustración 6-5: Costo tercera iteración

Cuarta iteración

Ilustración 6-6: Costo cuarta iteración

Quinta iteración

Ilustración 6-7: Costo quinta iteración

Coste total

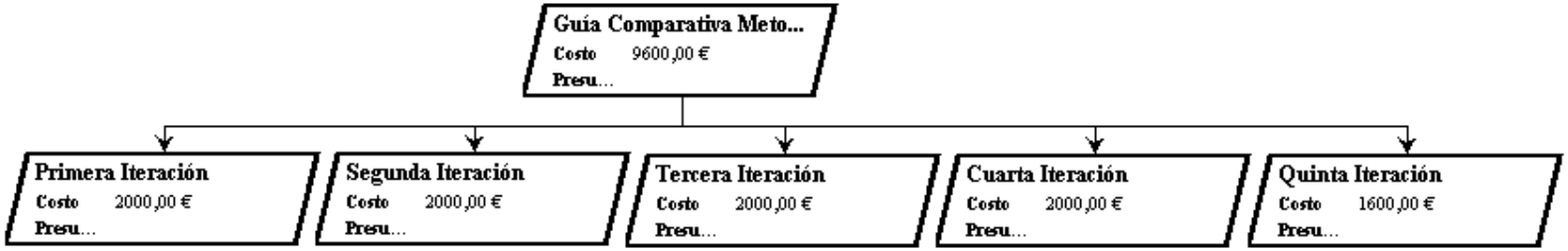


Ilustración 6-8: Costo total

Tanto en el desglose mensual como en el desglose por iteración el coste total del proyecto en recursos humanos es de: 9.600 €.

Recursos materiales

Componentes del presupuesto del proyecto que forman parte de la inversión inicial:

Hardware y Software

Recurso HW / SW	Coste Adquisición	% Coste aplicado proyecto	Coste Proyecto
TOSHIBA Satellite	700 €	20%	140 €
Windows XP Professional	170 €	20%	34 €
Microsoft-Office 2010	150 €	20%	30 €
Adobe Acrobat Professional	300 €	20%	60 €
Microsoft Paint	Integrado	0%	0 €
OpenProj	Open-source	0%	0 €
Consumibles	50 €	20%	10 €
Total			**274 €**

Tabla 6-4: Costo Hardware y Software

Comunicaciones

Recurso	Coste Unitario	Duración Meses	% Coste aplicado proyecto	Coste Proyecto
Conexión internet	20 €	3	20%	12 €
Total				**12 €**

Tabla 6-5: Costo Comunicaciones

El costo total invertido en recursos materiales es la suma del gasto en recursos Hardware + gasto recursos Software + gasto conexiones = 274 € + 12 € = **286 €**

Costo del proyecto

RECURSO	Coste Proyecto
HUMANOS	9.600 €
MATERIALES	286 €
TOTAL	**9.886 €**

Tabla 6-6: Costo total del proyecto

Herramientas Empleadas

Herramientas empleadas para el desarrollo del proyecto:

Hardware

- *TOSHIBA Satellite*: En Windows XP, Procesador Pentium 4 a 1,6 GHz. Máquina con la que se ha realizado todo el proyecto.

Software

- *Windows XP Professional*: Sistema operativo del ordenador utilizado para el desarrollo del proyecto
- *Microsoft-Office 2010.*
- *ScrumNinja*: Herramienta Online para la gestión de proyectos con Scrum.
- *OpenProj*: Herramienta open-source para la gestión de proyectos.
- *Adobe Acrobat Professional*.

Capítulo 7: Conclusiones

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Conclusiones

Para concluir este proyecto solo queda anotar algunas reflexiones sobre la realización de la guía comparativa de metodologías ágiles, en este proyecto se ha intentado dar una visión global sobre el mundo ágil, además de mostrar los principios de las metodologías ágiles más utilizadas. Se muestra como, compartiendo el mismo origen y los mismos principios, cada una de las metodologías ágiles estudiada tiene sus particularidades que hacen que se adapte mejor a un ámbito de trabajo u otro. Se ha creado una herramienta que permite conocer a una organización cuál es la metodología que mejor se adapta a modo de trabajo. Además se ha documentado un caso práctico de Scrum para comprender mejor como funciona en el día a día.

Las organizaciones tendrán un acceso rápido y clasificado de las metodologías estudiadas. Además, tener una herramienta que ayuda a elegir una metodología ágil entre varias creo que será muy importante para organizaciones que estén intentando instaurar una metodología ágil, partiendo de una organización tradicionalista, la herramienta permite clasificar a una organización dentro del mundo ágil. La organización se ahorra una gran cantidad de tiempo investigando las diferentes opciones, centrando sus esfuerzos en aplicar una metodología concreta y aumentando la probabilidad de éxito.

Se ha tratado de hacer un proyecto el objetivo claro de facilitar la vida a organizaciones interesadas en las metodologías ágiles, proporcionándoles una herramienta que les ayude a la toma de decisiones, que además tendrán una referencia teórica de cada metodología y una referencia práctica que les sirva de apoyo a la hora de comenzar a trabajar de manera ágil.

Como reflexión final creo que el proyecto en si es muy interesante y completo, da a conocer la esencia del mundo ágil, así como su impacto en las empresas, evitando situaciones de riesgo en los proyectos gracias a su adaptación a los cambios. Es importante tener en cuenta cómo la elección de una mala metodología de gestión de proyecto puede provocar un impacto negativo en el proyecto y a su vez en el cliente que ha solicitado el proyecto. Por otro lado, como es posible aplicar conjuntamente otra metodología ágil centrada en la mejora del desarrollo de software, creado productos de calidad. Me ha mostrado como es posible que un equipo trabaje cómodo y sea productivo y a la vez al cliente esté satisfecho.

Capítulo 8: Futuras líneas de trabajo

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Futuras líneas de trabajo

La guía comparativa de metodologías ágiles se ha centrado en un número limitado de metodologías, todas ellas pertenecientes al mundo ágil. Una posible ampliación podría ser añadir nuevas metodologías del agilismo o añadir también la metodología tradicional.

Otra posibilidad sería buscar nuevas características, propiedades de las metodologías que no se han contemplado en la herramienta creada y que pueden tener un peso importante a la hora de elegir una metodología u otra.

Capítulo 9: Glosario de Términos

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Glosario de Términos

Agilismo: Término que se usa en lugar del término anglosajón “Agile”. Se pueden utilizar los términos *desarrollo ágil de software*, *metodología ágil* ó *agilidad*.

Back-end: Parte de software que procesa la entrada desde el front-end.

Crontab: Administrador regular de procesos en segundo plano que ejecuta procesos a intervalos regulares de tiempo.

Front-end: Parte del software que interactúa con los usuarios.

Gráfica burndown: Gráfica que muestra la velocidad a la que se están completando los requisitos del sprint.

Historia de usuario: En las metodologías ágiles este término sustituye a la especificación de requisitos. Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos.

Kanban: Método ágil para la gestión de desarrollo software.

Manifiesto ágil: Recoge los principios en los que se basan todas las metodologías ágiles.

Metodología ágil: Métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requerimientos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios flexibles al cambio.

Product backlog: Lista de requisitos priorizada por el cliente.

Product owner: En equipos ágiles representa al cliente y es el encargado de negociar con el equipo.

Punto ideal: Unidad de medida ajustada del trabajo en un equipo ágil.

QA: Del anglosajón Quality Assurance, son las tareas, previas a la entrega al cliente, que se llevan a cabo para asegurar la calidad del producto software.

Refactorizar: Técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Scrum: Método ágil para la gestión de desarrollo software.

Scrumban: Método ágil para la gestión de desarrollo software.

Scrum daily meeting: Reunión diaria de Scrum en la que participan todos los miembros del equipo y comentan el progreso del trabajo y posibles bloqueos que hayan surgido en el trabajo.

Scrum master o facilitador: En equipos ágiles responsable de guiar y resolver obstáculos al equipo.

Sprint o iteración: Ritmo de los ciclos de Scrum. Está delimitado por la reunión de planificación del sprint y la reunión retrospectiva.

Sprint backlog: Lista priorizada con los requisitos seleccionados para un Sprint.

Sprint demonstration o demo: Reunión con el cliente, propia de Scrum, en la que se muestran las nuevas funcionalidades.

Sprint restrospective o retrospectiva: Reunión propia de Scrum, en la que se reflexiona sobre el Sprint que acaba de finalizar y se determina qué podría cambiarse para el siguiente Sprint y ser más productivo y mejor.

Velocidad del equipo o velocidad de trabajo: Número de puntos ideales de trabajo que es capaz de asumir el equipo en el presente Sprint.

XP: Método ágil para la gestión de desarrollo software.

Capítulo 10: Bibliografía

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Bibliografía

The Agile Samurai: How Agile Masters Deliver Great Software, Jonathan Rasmusson

SCRUM and XP from the Trenches (Enterprise Software Development), Henrik Kniberg

Do Better SCRUM, artículo escrito por Certified Scrum Coach and Trainer Peter Hundermark

Capítulo 11: Referencias

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Referencias

http://www.eumed.net/libros/2009c/584/indice.htm

http://agilemanifesto.org/

http://www.scrumalliance.org

http://www.programacionextrema.org/

http://www.extremeprogramming.org/rules.html

http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

http://www.willydev.net/descargas/masyxp.pdf

http://www.agileproductdesign.com/blog/2009/kanban_over_simplified.html

http://ceur-ws.org/Vol-341/paper9.pdf

http://www.scrumninja.com/scrum-software

Capítulo 12: Índice de Tablas

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Índice de Tablas

Capítulo 13: Índice de Figuras

María José Pérez Pérez
Grado en Ingeniería Informática de Servicios y Aplicaciones

Índice de figuras