

Implantación de la metodología ágil Kanban en un equipo de desarrollo de software



Diego J. Romero López

Trabajo Fin de Máster

Director: Dr. Agustín Yagüe Panadero

Máster en Dirección y Gestión de Proyectos Software 2016

Universidad Politécnica de Madrid

Índice

1 Agradecimientos.....	8
2 Introducción.....	9
2.1 Sobre mí.....	9
2.2 ¿Qué contiene este documento?.....	9
3 Objetivo.....	11
4 Estado actual del equipo y retos a superar.....	12
4.1 Proyectos ya iniciados.....	12
4.2 Distinto origen de cada uno de los desarrolladores.....	12
4.3 Proyectos heterogéneos.....	13
4.4 Clientes diferentes.....	14
4.5 Varios proyectos por desarrollador.....	14
4.6 Propiedad individual del código.....	15
4.7 Desarrollo y mantenimiento en paralelo.....	15
4.8 Proyectos con alta deuda técnica.....	15
4.9 Proyectos con baja usabilidad.....	15
4.10 Facturación por horas.....	16
4.11 Inexistencia de práctica ágil.....	16
4.12 Entorno.....	17
4.13 Falta de estándares y cultura de desarrollo software con calidad.....	18
4.14 Dificultades para el paso a producción del software.....	18
5 Prácticas ágiles seleccionadas.....	19
5.1 Kanban.....	19
5.1.1 Introducción.....	19
5.1.2 ¿Por qué Kanban?.....	19
5.1.2.1 Limitación de multitarea.....	19
5.1.2.2 Integración con el método Pomodoro.....	19
5.1.2.3 Desarrollo y mantenimiento.....	21
5.1.2.4 De fácil implantación.....	21
5.1.2.5 Proyectos con equipo cambiante.....	21
5.1.2.6 Kanban no obliga a olvidar el resto de prácticas ágiles.....	21
5.2 Autoorganización.....	21
5.3 El foco de un desarrollador ha de ser el desarrollo.....	22
5.4 Código de calidad.....	22
5.4.1 Minimización del estado global.....	23
5.4.2 Orientación a objetos.....	23
5.4.3 Principio de única responsabilidad.....	24
5.4.4 Nombres de variables.....	24
5.4.5 Reducción de la complejidad ciclomática.....	24
5.4.6 Refactorización.....	24
5.4.7 Revisiones de código.....	24
5.4.8 Programación por parejas.....	25
5.5 Gestión del tiempo.....	25
5.6 Entregas frecuentes.....	25
5.7 Colaboración con el cliente.....	25
5.8 Documentación.....	26
5.9 Pruebas e integración continua.....	26

5.10 Dar valor al software.....	26
5.11 Otras buenas prácticas.....	27
5.11.1 Formación continua.....	27
5.11.2 Autoevaluación.....	27
5.11.3 Potenciar estudios de usabilidad.....	27
6 Metodología de trabajo.....	28
6.1 Introducción.....	28
6.2 Estructura del tablero Kanban.....	28
6.2.1 Estados de las tareas.....	28
6.2.2 Límite de <i>Work in Progress</i> (WIP).....	29
6.3 Herramientas software.....	29
6.3.1 Trello.....	29
6.3.2 Plus for Trello.....	31
6.4 Prácticas ágiles no canónicas de Kanban.....	31
6.4.1 Reuniones diarias.....	31
6.4.2 Reuniones de retrospectiva.....	31
6.4.3 Colaboración y reuniones con dueños del producto.....	32
6.4.4 Colaboraciones con otros desarrolladores.....	34
7 Aplicación de los estudios de usabilidad al proceso de desarrollo.....	35
7.1 Introducción.....	35
7.2 Formación.....	35
7.3 Modelo iterativo de usabilidad.....	35
7.3.1 Bocetos.....	36
7.3.2 Evaluaciones heurísticas.....	36
7.3.3 Búsqueda de soluciones.....	36
7.3.4 Evitar implementación de “Dark patterns”	37
7.3.5 Tests de usabilidad.....	39
7.3.6 Accesibilidad.....	39
7.3.7 Tests sobre usuarios reales.....	40
7.4 Herramientas.....	40
7.4.1 Balsamiq.....	40
7.4.2 Evaluadores de accesibilidad.....	41
7.4.3 Herramientas de escritorio remoto.....	42
8 Planificación.....	43
8.1 Primer mes: Kanban y Pomodoro.....	43
8.2 Segundo mes: tests y revisiones de código.....	43
8.3 Tercer mes: usabilidad.....	44
8.4 Cuarto mes: mejorar la comunicación con los clientes y final.....	44
8.5 Futuro.....	45
9 Evaluación.....	46
9.1 Introducción.....	46
9.2 Evaluación de la implantación.....	46
9.3 Evaluación de la mejora alcanzada.....	49
9.3.1 Métricas.....	49
9.3.1.1 Lead.....	49
9.3.1.2 Cycle.....	49
9.3.1.3 Tiempo medio por estado.....	49
9.3.1.4 Número de vueltas atrás por estado.....	50

9.3.1.5 Tiempo estimado de desarrollo.....	50
9.3.1.6 Tiempo efectivo de desarrollo.....	51
9.3.1.7 Velocidad de desarrollo semanal.....	51
9.3.1.8 Número de tareas por hora.....	51
9.3.1.9 Número de incidencias por proyecto.....	52
9.3.2 Herramientas para realizar las mediciones.....	52
9.3.3 Mediciones subjetivas.....	55
9.3.3.1 Número aproximado de interrupciones.....	55
9.3.3.2 Cuestionarios.....	56
9.3.3.2.1 Dirección.....	56
9.3.3.2.2 Equipo comercial y clientes.....	58
9.3.3.2.3 Desarrolladores del equipo.....	59
9.3.3.2.4 Mi bitácora.....	61
10 Resultados de la primera evaluación del proceso: implantación de Kanban.....	62
10.1 Introducción.....	62
10.2 Resultados.....	63
10.2.1 Proyecto A.....	63
10.2.2 Proyecto B.....	64
10.2.3 Proyecto C.....	65
10.2.4 Proyecto D.....	66
10.2.5 Proyecto E.....	67
10.2.6 Proyecto F.....	68
10.3 Conclusiones.....	70
11 Resultados de la segunda evaluación del proceso: cuestionarios de evaluación del equipo y del proceso.....	72
11.1 Dirección.....	72
11.1.1 Preguntas numéricas.....	72
11.1.2 Preguntas abiertas.....	72
11.1.2.1 ¿Qué te gustaría mejorar del manager?.....	72
11.1.2.2 ¿Qué te gustaría mejorar del equipo?.....	73
11.1.2.3 Comentarios adicionales.....	73
11.1.3 Análisis sobre los resultados.....	73
11.2 Dueño del producto.....	74
11.2.1 Preguntas numéricas.....	74
11.2.2 Preguntas abiertas.....	74
11.2.2.1 ¿Qué te gustaría mejorar del equipo?.....	74
11.2.2.2 Comentarios adicionales.....	74
11.2.3 Análisis sobre los resultados.....	74
11.3 Equipo.....	75
11.3.1 Preguntas numéricas.....	76
11.3.2 Preguntas abiertas.....	77
11.3.2.1 ¿Qué te gustaría mejorar del facilitador?.....	77
11.3.2.2 ¿Qué te gustaría mejorar del equipo?.....	77
11.3.2.3 ¿Qué te gustaría mejorar del entorno?.....	77
11.3.2.4 Comentarios adicionales.....	77
11.3.3 Análisis sobre los resultados.....	77
12 Evaluación de las métricas de Kanban por proyecto.....	80
12.1 Proyecto A.....	81

12.1.1 Mediciones.....	81
12.1.1.1 Tiempo medio de las tareas en cada estado.....	81
12.1.1.2 Lead.....	82
12.1.1.3 Cycle.....	82
12.1.1.4 Estados que ocasionan vueltas atrás.....	82
12.1.1.5 Tiempos de desarrollo.....	82
12.1.2 Análisis de los resultados.....	82
12.2 Proyecto B.....	83
12.2.1 Mediciones.....	83
12.2.1.1 Tiempo medio de las tareas en cada estado.....	83
12.2.1.2 Lead.....	84
12.2.1.3 Cycle.....	84
12.2.1.4 Estados que ocasionan vueltas atrás.....	85
12.2.1.5 Usuarios que generan vueltas atrás.....	86
12.2.1.6 Tiempos de desarrollo.....	86
12.2.2 Análisis de los resultados.....	86
12.3 Proyecto C.....	87
12.3.1 Mediciones.....	87
12.3.1.1 Tiempo medio de las tareas en cada estado.....	87
12.3.1.2 Lead.....	87
12.3.1.3 Cycle.....	88
12.3.1.4 Estados que ocasionan vueltas atrás.....	88
12.3.1.5 Tiempos de desarrollo.....	88
12.3.2 Análisis de los resultados.....	88
12.4 Proyecto D.....	90
12.4.1 Mediciones.....	90
12.4.1.1 Tiempo medio de las tareas en cada estado.....	90
12.4.1.2 Lead.....	90
12.4.1.3 Cycle.....	90
12.4.1.4 Estados que ocasionan vueltas atrás.....	91
12.4.1.5 Tiempos de desarrollo.....	91
12.4.2 Análisis de los resultados.....	91
12.5 Proyecto E.....	92
12.5.1 Mediciones.....	92
12.5.1.1 Tiempo medio de las tareas en cada estado.....	92
12.5.1.2 Lead.....	92
12.5.1.3 Cycle.....	92
12.5.1.4 Estados que ocasionan vueltas atrás.....	93
12.5.1.5 Tiempos de desarrollo.....	93
12.5.2 Análisis de los resultados.....	93
12.6 Proyecto F.....	94
12.6.1 Mediciones.....	94
12.6.1.1 Tiempo medio de las tareas en cada estado.....	94
12.6.1.2 Lead.....	94
12.6.1.3 Cycle.....	95
12.6.1.4 Estados que ocasionan vueltas atrás.....	95
12.6.1.5 Tiempos de desarrollo.....	95
12.6.2 Análisis de los resultados.....	95

12.7 Conclusiones.....	96
13 Evaluación personal del proceso de mejora.....	97
14 Conclusiones finales.....	99
14.1 Sobre la aplicación del proceso de mejora Kanban.....	99
14.1.1 Los cambios en organizaciones tardan tiempo.....	99
14.1.2 Siempre hay gente que se resiste a un cambio.....	99
14.1.3 La calidad software ha de ser lo primero.....	99
14.1.4 La formación del equipo es fundamental.....	100
14.1.5 La motivación es parte importante del proceso.....	100
14.1.6 El objetivo es la mejora constante: el <i>kaizen</i>	100
14.1.7 Kanban no es sólo usar un “tablero de tareas”.....	101
14.1.8 Rendimiento del trabajo.....	101
14.1.9 Métricas de Kanban.....	101
14.1.10 Midiendo el desperdicio.....	101
14.1.11 Políticas con los equipos con los que colaboramos.....	102
14.1.12 Colaboración con clientes.....	102
14.2 Personales.....	102
14.2.1 Investigación en ingeniería del software real.....	102
14.2.2 La gestión de proyecto software es compleja.....	103
14.2.3 Un reto a superar.....	103
15 Bibliografía y recursos.....	104
16 Anexo 1: cuestionario de evaluación de implantación Kanban de Christophe Achouiantz.....	106
17 Anexo 2: manual de usuario y resultados de ejemplo de PyStats-Trello.....	107
17.1 Fichero Readme.....	107
17.2 Fichero de salida de ejemplo.....	109

1 Agradecimientos

Agradezco a intelligenia el darme esta oportunidad de iniciar este proceso de mejora y poder usarlo en mi formación. En especial agradezco al director ejecutivo de esta empresa, José Carlos Calvo Tudela el hacernos sentir parte integrante de la misma.

También me gustaría agradecer a los miembros de mi equipo de trabajo: Brian, José Miguel, Franciscoy Salvador. Sin ellos esto no hubiera sido posible.

Por supuesto, agradezco a todos los profesores de este máster en Dirección y Gestión de Proyectos Software su trabajo y todo el esfuerzo que han hecho al ayudarnos en hacer de mí un mejor profesional en el mundo de la Ingeniería del Software.

Por último, agradezco a mi familia y a Raquel todo el apoyo que me han dado durante todo este tiempo.

2 Introducción

2.1 Sobre mí

Soy Diego J. Romero López, subdirector técnico en [intelligenia](#), una pequeña empresa de Granada (España) que realiza desarrollos de aplicaciones web y de teléfonos móviles. Desde que obtuve el título de ingeniero informático, toda mi vida profesional ha transcurrido en el departamento técnico de esta empresa.

Comencé como desarrollador de aplicaciones web y conforme fue pasando el tiempo, fui creciendo en el ámbito profesional y técnico. Ahora, era el momento de hacerlo en el ámbito de la gestión de proyectos.

2.2 ¿Qué contiene este documento?

Este documento describe el proceso de mejora que he aplicado a un equipo de desarrollo liderado por mí para su conversión en un equipo ágil.

He estructurado este documento en las siguientes secciones:

En la sección 3 vamos a describir el objetivo de este trabajo fin de máster. ¿Qué es lo que busco con este trabajo? ¿Por qué he propuesto este proceso de mejora?

Ya en la sección 4, describimos la naturaleza del equipo y sus proyectos. El equipo tiene sus particularidades y los proyectos tienen también sus dificultades por diversos motivos que se describen en dicha sección.

Después, en la sección 5 me voy a centrar en qué prácticas ágiles quiero aplicar al equipo. Además, incluiré algunas prácticas adicionales como son la formación continua o aplicar estudios de usabilidad al desarrollo. Hacer interfaces usables ha de ser una prioridad para facilitar que los usuarios comprendan cómo usar nuestras aplicaciones web sin necesidad de consultarnos.

En la sección 6 seguiré con la metodología de trabajo a seguir, el uso de Kanban, así como las herramientas software y prácticas ágiles que sostendrán este proceso.

La sección 7 contiene las prácticas para mejorar la usabilidad. Siendo como es un campo tan extenso, he preferido centrarme en definir un modelo iterativo de mejora de la usabilidad en nuestras aplicaciones, perfecto para nuestro enfoque ágil.

La sección 8 contiene la planificación que voy a seguir a lo largo de todo este proceso. Desde el día 15 de Febrero hasta el día 15 de Mayo del año 2016.

No sería un proceso completo si no pudiera compararlo con la forma de trabajar anterior. Para ello, en la sección 9, describo varias evaluaciones que se van a hacer desde finales de Abril a mediados de Mayo (en función de cómo transcurra el calendario).

Las secciones 10, 11, 12 y 13 son los resultados de las evaluaciones. Primero, en la sección 10 se

muestran los resultados de la calidad de la implantación de Kanban. En cambio, en la sección 11 se muestran los resultados de una evaluación mediante cuestionarios a distintos implicados en los proyectos. Para cerrar las evaluaciones sobre los proyectos, en la sección 12, evaluamos las distintas métricas de Kanban para cada uno de los proyectos para conocer en qué estamos fallando y cómo podemos continuar con el proceso de mejora. Estas tres secciones contienen, además de los propios resultados, un análisis de los mismos. Por último, la sección 13 contendrá una evaluación personal sobre el proceso basándome en las notas que he ido tomando a lo largo de éste.

La última sección, la 14 contiene las conclusiones de todo este proceso. Desde mi opinión personal a las enseñanzas que he aprendido en este camino que acabo de comenzar.

3 Objetivo

A mediados de Febrero del 2016 la dirección de intelligenia me encomendó la tarea de liderar de forma efectiva un equipo de desarrolladores.

El director ejecutivo espera que mejore el rendimiento en su desarrollo en cada uno de sus proyectos y que incremente la calidad del software generado. Así, obtendremos productos de mayor calidad de forma más eficientemente y esto redundará en una mayor satisfacción de los clientes.

Dado que la dirección confía en mí, mi objetivo principal es **mejorar el rendimiento del equipo mediante la aplicación de metodologías ágiles y buenas prácticas de desarrollo**.

Desde el año 2008 hemos estado usando un marco de trabajo tradicional y eso nos ha hecho tener bastantes proyectos con pérdidas y varios clientes descontentos, al no ser capaces de sintetizar en un documento sus ideas sobre el proyecto.

Quiero cambiar la forma de trabajar que tenemos y conseguir que los clientes estén satisfechos con el tiempo de respuesta a los cambios solicitados y con el resultado final de sus proyectos.

Así que este trabajo de fin de máster no es una simulación de una serie de iteraciones de desarrollo ágil: es la descripción del proceso de implantación de Kanban en un equipo real de desarrollo software.

4 Estado actual del equipo y retos a superar

El equipo que voy a gestionar es un equipo con unas características bastante peculiares que hacen la implantación de las metodologías ágiles sea algo compleja.

Enumeramos los problemas que hay actualmente en el equipo:

4.1 Proyectos ya iniciados

Los proyectos que sobre los que vamos a trabajar ya han sido comenzados. Esto es, se van a hacer labores de mantenimiento que requerirán mejoras puntuales al software.

Esto puede parecer una ventaja puesto que un software implantado y en funcionamiento es más fácil de tratar, pero nos enfrentamos a:

- Desconocimiento de la lógica de negocio.
- Desconfianza del cliente por el cambio de equipo.
- Proyectos en pérdidas: varios de los proyectos que se nos han asignados fueron presupuestados de una forma inadecuada y entraron en sobrecoste antes de que nos los asignaran. Así, nuestra tarea también será la de educar al cliente en la complejidad de las estimaciones software y en la minimización de las pérdidas.

4.2 Distinto origen de cada uno de los desarrolladores

Cada uno de los integrantes de mi equipo tiene distinto nivel de experiencia y conocimientos. Excluyéndome a mí, sólo hay un graduado en Ingeniería Informática, siendo el resto de desarrolladores titulados en ciclos formativos superiores.

La formación de los titulados en ciclos superiores es mucho menor que la de los ingenieros, careciendo de conocimientos en:

- Fundamentos matemáticos.
- Informática teórica.
- Algorítmica.
- Ingeniería del Software.
- Arquitectura hardware.
- Redes de computadores.
- Optimización de software.
- Planificación.

Esto supondrá que determinadas tareas no podrán hacerlas sin una supervisión por alguno de los dos

ingenieros informáticos del equipo.



De izquierda a derecha: Brian, José Miguel, yo, Francisco y Salvador. Nuestro equipo al completo. José Miguel es casi graduado (sólo le falta de entregar el trabajo fin de grado) y el resto (salvo yo) son titulados en ciclos formativos de grado superior.

4.3 Proyectos heterogéneos

Cada uno de los proyectos que tienen los desarrolladores tiene su propia idiosincrasia y recorrido tecnológico.

Para empezar, algunos proyectos comenzaron en el año 2008, mientras que otros tienen un par de años o, incluso, no están terminados.

Las tecnologías son variadas. Hay proyectos desarrollados con PHP (con distintos *frameworks* propietarios), mientras que hay otros desarrollados con Python y Django.

Los proyectos desarrollados con Python y Django sí están documentados y siguen las convenciones de este *framework*. En cambio, los desarrollados con PHP se basan en un *framework* propio y necesitarán un proceso de refactorización y documentación para poder desarrollar software de calidad.

Concretamente, se nos han asignado una serie de proyectos que identificaremos con una letra del abecedario por motivos de salvaguardar la privacidad de la empresa y de nuestros clientes:

- **A:** Aplicación web con interfaz de servicios web de terceros (PHP + Python + Django).
- **B:** Aplicación web de gamificación (Python + Django).
- **C:** Gestión de eventos (PHP con *framework* propio).

- **D:** Mantenimiento de contenidos y maquetación de un portal para un cliente (OpenCMS).
- **E:** Maquetación de la web de nuestra empresa (HTML + CSS + Javascript).
- **F:** Aplicación web de venta de entradas (Python + Django).
- **G:** Aplicación web de gestión turística (Python + Django).
- **H:** Adición de funcionalidad a una plataforma web de nuestro cliente para proporcionar interconexión con un tercero (PHP).
- **I:** portal con estadísticas, registros e interfaces de A (Python + Django).

4.4 Clientes diferentes

La tipología del cliente es distinta en cada caso.

En primer lugar, la naturaleza del cliente es distinta: los proyectos de venta de entradas y gestión de eventos tienen como cliente al departamento comercial, ya que éste es el encargado de (a su vez) vender el producto a un tercero. El resto de proyectos tiene como clientes a otras empresas.

Después, la persona que nos sirve como enlace, el dueño del producto puede tener mayor o menor grado de implicación en el proyecto. Además, éste puede no tener muchos conocimientos técnicos (o incluso sobre su negocio).

Por ejemplo, en el proyecto A el dueño del proyecto es una persona que participa activamente en el proceso de desarrollo: revisa las entregas y participa en las reuniones para determinar la funcionalidad o planificación del proyecto. Su actitud hace que el desarrollo sea fluido y se consigan entregas muy próximas a sus necesidades.

En cambio, en el proyecto G, nuestro enlace con la empresa es una persona de conocimientos limitados y que suele faltar a las formas. Tampoco muestra mucha disponibilidad ni interés con el proyecto. Su actitud daña la motivación de los miembros del equipo, y en general, también daña al desarrollo del proyecto.

Evidentemente, implantar un enfoque ágil dependerá mucho del nivel de compromiso que quiera alcanzar el dueño del producto. Si éste no tiene interés o hace las cosas más difíciles al equipo de desarrollo, no podemos trabajar con él con un enfoque totalmente ágil.

4.5 Varios proyectos por desarrollador

Cada desarrollador tiene varios proyectos para distintos clientes. Esto hace que el cambio de contexto sea más grave si acaso, ya que no sólo se cambia de tarea, sino de proyecto. Siendo cada proyecto de una naturaleza completamente distinta.

Como hemos visto, algunos proyectos son de distinta tecnología y paradigma, lo que hace aún si cabe más complicado el pasar de desarrollar tareas de uno a otro.

Deberemos aprender a gestionar incidencias de varios proyectos a la vez, evaluando su prioridad, no

en función de la importancia que nos indique el dueño del producto, sino en función de su importancia real.

4.6 Propiedad individual del código

Ahora mismo el conocimiento de cada proyecto reside en el desarrollador que ha hecho el proyecto. Esto hace prácticamente imposible la sustitución de éste por otro desarrollador de forma sencilla. Siendo obligatorio hacer pasar al nuevo desarrollador por un proceso de formación.

En el caso de los proyectos que se nos han asignado de otros equipos, la situación es más dramática aún, debido a la falta de documentación técnica. Esto hace que en al menos un proyecto, el mantenimiento sea también una tarea de “investigación”.

4.7 Desarrollo y mantenimiento en paralelo

Otra de las cuestiones que hacen que este equipo sea peculiar, es el hecho de que muchas de las tareas que se han de resolver son de mantenimiento.

De esta manera, cada uno de nosotros ha de conjugar las tareas de desarrollo con las incidencias, e incluso el soporte técnico al departamento comercial (o usuarios finales).

Como se puede imaginar, esto hace que el desarrollo pueda ser interrumpido por una incidencia que necesita ser resuelta. De hecho, puede que esa incidencia requiera un trabajo de varias horas, deteniendo el transcurso de un proyecto durante el desarrollo de dicha incidencia.

4.8 Proyectos con alta deuda técnica

Ya hemos comentado que dentro de los proyectos que tiene que mantener mi equipo, hay proyectos sin documentación. Ni sobre el análisis ni sobre el diseño software.

Para empeorar aún más la situación, las prácticas de codificación del equipo anterior no eran las adecuadas, y la carencia de estructura y comentarios de calidad en el código es bastante frecuente.

Se ha planteado a la dirección la posibilidad de una refactorización de algunas partes del código, pero ha de estudiarse desde el punto de vista financiero, por lo que a día de hoy, no es una opción viable.

4.9 Proyectos con baja usabilidad

No se ha hecho en ningún proyecto un estudio de usabilidad. Esto ha hecho que en algunos proyectos los clientes no sepan realizar las acciones y tengan que contactar con el soporte telefónico de la empresa.

Las interfaces y conceptos que usamos son nativos del proyecto y es complicado que los usuarios sean capaz de aprender nuevos conceptos. El software no es sencillo de usar ni evidente, lo que dificulta el aprendizaje y autonomía de los clientes a la hora de usarlo.

Por otro lado, tampoco hay manuales de usuario bien redactados y los pocos que hay no se los leen

los clientes, por lo que es como si no los hubiera.

En definitiva, esto hace que el número de incidencias en algunos proyectos sea muy alto no por fallos del software, sino porque los usuarios no saben usarlo o no entienden los conceptos, por lo que se ven incapaces de aprovecharlo al 100% o de realizar algunas tareas¹.

4.10 Facturación por horas

Todos los proyectos en los que trabajamos (salvo dos) facturan al cliente por horas. Esto nos permite una flexibilidad máxima a la hora de ejecutar las mejoras en sus proyectos software.

Normalmente se da una estimación para las tareas de mejora o ampliación que ha solicitado el cliente y (si las aprueba), se comienza la mejora.

Cada mes se le pasa una factura al cliente con las horas consumidas. Nótese que dado que la cultura de los clientes en cuanto a proyectos software es muy baja, no se cobran los análisis ni diseños de proyectos ni el soporte técnico. Esto hace que las horas de desarrollo consumidas por el equipo tengan que ser ponderadas por un factor. Este factor dependerá del desarrollador, del tipo de proyecto y de otros factores. Este factor permite tener cierto margen de beneficios más allá de cubrir simplemente costes.

A continuación, mostramos una hoja de cálculo con la cuenta de las horas de desarrollo de cada funcionalidad y comentarios adicionales.

The screenshot shows a Google Sheets spreadsheet with the following structure:

	A Fecha	B Horas	C Tarea	D Autor	E Comentarios	F	G	H
1	Fecha	Horas	Tarea	Autor	Comentarios			
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								

Según lo acordado con el cliente, esta hoja puede compartirse con el cliente o directamente enviarle cada mes una selección de las tareas realizadas (y su coste) durante ese período de tiempo.

4.11 Inexistencia de práctica ágiles

En la empresa en la que estoy, nunca se han implantado prácticas ágiles. Se ha seguido un enfoque

¹ Es cierto que esto varía según la cultura del cliente, pero no podemos confiar en tener siempre clientes con conocimientos técnicos (o ganas de aprender a usar la plataforma).

tradicional con una etapa de extracción de requisitos, luego un diseño y por último la codificación.

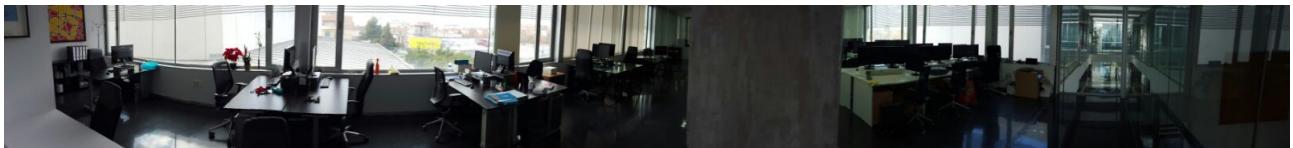
Se mejoró este proceso con la introducción de un proceso iterativo basado en *storyboards*. Desde hace unos años, para todo proyecto nuevo, se desarrollan bocetos para todas las pantallas y se validan con el cliente una y otra vez. Así, sólo se comienza el desarrollo cuando el cliente ha validado completamente todas las pantallas de su aplicación web, además del flujo de ejecución entre ellas.

Como se puede imaginar, esto ha ayudado mucho en el desarrollo de los proyectos y a evitar discrepancias con los requisitos.

Más allá de usar este proceso iterativo de *storyboarding*, no se ha llevado a la práctica ninguna característica adicional de desarrollo ágil.

4.12 Entorno

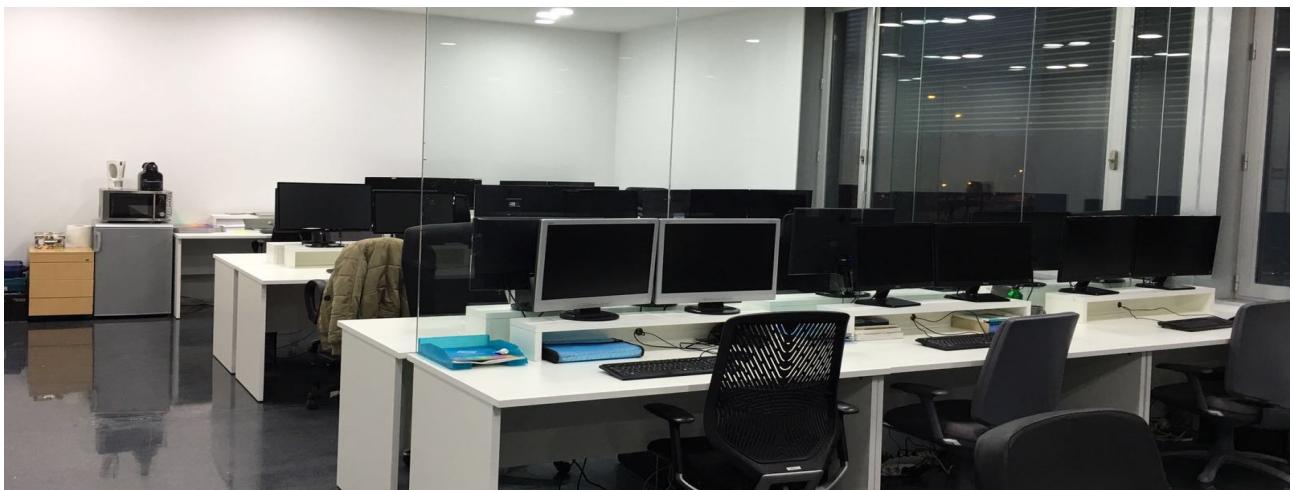
El entorno no es el mejor entorno para desarrollar software. La zona comercial en la que se trata con clientes y se contesta al teléfono no tiene separación física en uno de sus extremos (comparten la puerta), lo que hace que se emita un ruido constante a nuestra zona de desarrollo:



Panorámica de la zona comercial (izquierda) y zona de desarrollo (derecha) desde la puerta de la oficina de intelligenia. (c) intelligenia.

Los comerciales tienden a acercarse al puesto de trabajo de los desarrolladores e interrumpirlos con incidencias, dudas o cuestiones.

Por otro lado, la zona de desarrollo sólo tiene separaciones entre las mesas mediante un cristal, dejando un pasillo abierto en la zona izquierda:



Zona de desarrollo. Se pueden ver los cristales entre los grupos de mesas y la zona abierta a la izquierda que se usa como pasillo. (c) intelligenia.

Esta mínima separación hace que el ruido se transmita sin problemas por la zona de desarrollo. Por ejemplo, cuando un grupo de más de dos personas está hablando, la conversación se escucha fácilmente desde cualquier punto de la oficina.

También, al ser zona de paso, el pasillo es una zona constante de generación de ruido. Esto supondrá un reto para aquellos que tenemos el puesto en el lado izquierdo (como es mi caso).

4.13 Falta de estándares y cultura de desarrollo software con calidad

En intelligenia hay equipos que no usan UML o no hacen documentos de análisis, lo que hace que estudiar uno de sus proyectos sea una tarea muy compleja puesto que no hay forma de conocer la funcionalidad completa del proyecto.

Por otro lado, hay equipos que no tienen buenas prácticas de programación. Esto dificulta de sobremanera el mantenimiento y la mejora de sus proyectos puesto que manejar ficheros de gran tamaño o sin abstracción es muy complejo.

En definitiva, bien por un motivo u otro, hay equipos que generan software con alta deuda técnica debido a que no existe una política uniforme de calidad software.

Esto supone un reto porque hay que armonizar desarrolladores ya no de diferentes niveles formativos, sino con distintas costumbres, niveles de calidad y estándares de desarrollo.

4.14 Dificultades para el paso a producción del software

En la empresa el paso a producción de las distintas revisiones de código para uno de los proyectos los realiza un equipo de administradores de sistemas.

El mecanismo para solicitar una actualización de un servidor es una hoja de cálculo en la que se indica una URL de un proyecto, una revisión y una prioridad. Si la prioridad es urgente se envía de forma automática un correo a los administradores de sistemas para notificarles la necesidad de realizar la actualización.

Estos administradores de sistemas además de realizar esta tarea también desarrollan. Como se puede imaginar, esto hace que una actualización pueda tardar varias horas en producirse, y para colmo de males, no hay un plazo máximo acordado en el que se tenga que realizar.

Esta forma de trabajar supone un problema debido a la existencia de proyectos sobre los que no se pueden hacer pruebas completas debido a su alta deuda técnica y a la falta de los datos reales. Esto es, no hay forma de probar determinadas funcionalidades más que en el servidor de explotación. Si se actualiza una aplicación cuando el desarrollador principal no está presente y ocurre un error, pocos desarrolladores (puede que nadie) sabrán arreglarlo.

5 Prácticas ágiles seleccionadas

Dado que el equipo necesita un cambio global, planteamos una serie de buenas prácticas que serán las que hagan mejorar el trabajo. Además de las prácticas ágiles, se incluyen buenas prácticas de desarrollo de software y de usabilidad.

5.1 Kanban

5.1.1 Introducción

Kanban es un sistema de planificación de la producción de origen japonés implantado en 1953.

El origen de este método era el ser más eficientes en la gestión de recursos en el proceso de fabricación de vehículos en Toyota. Kanban les servía para sincronizar las necesidades de inventario con el consumo real de productos necesarios para mantener una producción en la factoría.

Inspirado en este método y por el sistema de fabricación Lean [LEAN] David Anderson definió de manera más formal el método Kanban para su aplicación en desarrollo de software [ANDERS10]. Usaremos el texto [ANDERS10] como base a lo largo de este proceso de mejora.

Este método está basado en la visualización del flujo de trabajo de cada una de las tareas en un tablero. En este tablero las columnas representan los estados y las tareas avanzan o retroceden en función del estado que alcancen.

Los principios básicos de Kanban son los siguientes:

- Comienza el proceso de mejora con los procesos que ya existen.
- Persigue el cambio y la mejora incremental.
- Respeta el proceso, los roles, responsabilidades y títulos actuales.
- Liderazgo constante a todos los niveles.

Como se puede ver, no es sólo el usar un tablero, sino un cambio de cultura de “desarrollo tradicional” a “desarrollo ágil”, basándose en varias herramientas, entre ellas, un tablero de tareas.

5.1.2 ¿Por qué Kanban?

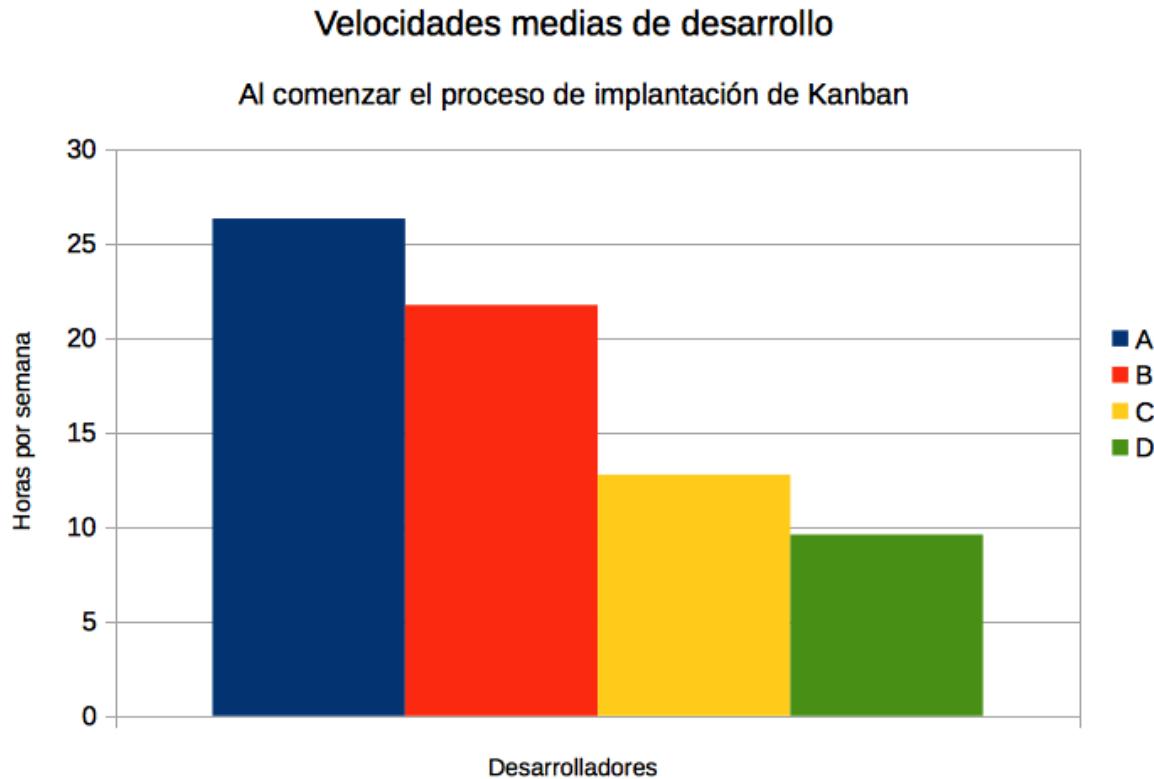
5.1.2.1 Limitación de multitarea

Una de las prácticas ágiles que comenté más adelante es la de concentración. Mediante la definición de un máximo *WIP* (*Work in Progress*) en las columnas de desarrollo, se limita la multitarea de los miembros de mi equipo.

5.1.2.2 Integración con el método Pomodoro

El concepto de tarea permite aplicar el método Pomodoro a cada una de las tareas del tablero. Al comenzar con este proceso de mejora les pedí a cada uno de los miembros de mi equipo que registraran las velocidades de desarrollo durante las cuatro primeras semanas.

A continuación muestro los resultados:



Actualmente nuestra semana tiene 39 horas y como se puede ver el desarrollador que más tiempo ocupa al desarrollo llega a unas 26 horas a la semana. Esto es, el máximo tiempo de desarrollo es sólo de un 66%. Este desarrollador sufría pocas interrupciones y se acercaba bastante al máximo teórico del 70% del tiempo desarrollando.

El desarrollador B no se planificaba correctamente las tareas y no lograba más de 22 horas de desarrollo a la semana.

En cambio, el desarrollador C sufría interrupciones constantes del departamento comercial, lo que hacía que no lograse la concentración suficiente para poder tener una eficiencia aceptable.

El desarrollador D no tenía suficiente experiencia en el desarrollo con la tecnología que estamos usando en aquél momento, por lo que su rendimiento es normal que sea tan bajo al estar en un proceso de formación.

Si bien es cierto que la velocidad de desarrollo no es una métrica adecuada para medir el rendimiento (de forma aislada), si es cierto que es una forma de medir el tiempo que dedican los desarrolladores al desarrollo de software. En este caso, estas mediciones hicieron que nos decantásemos por un método que promoviese las cajas de tiempo.

El resto de problemas de interrupciones y falta de concentración lo tratamos más adelante en esta

misma sección.

5.1.2.3 Desarrollo y mantenimiento

Debido a que los desarrolladores se dedican a resolver incidencias y desarrollar mejoras sobre proyectos ya implantados, hemos optado por dividir el trabajo en tareas y usar Kanban para ello.

Scrum es un marco de trabajo muy útil para el desarrollo de nuevos proyectos. En nuestro caso sólo un proyecto es un desarrollo nuevo (y ya tenía el análisis completamente parcialmente hecho cuando nos lo entregaron). El resto de proyectos generan incidencias con alta prioridad por lo que no se adaptan a un enfoque Scrum.

5.1.2.4 De fácil implantación

Kanban es un método de mejora continua que tiene pocas normas y eso permite que de forma sencilla se pueda implantar en un equipo con bajos conocimientos de agilidad en el mundo del desarrollo software.

5.1.2.5 Proyectos con equipo cambiante

Otro problema que tiene implantar Scrum es que, en primer lugar, el hecho de que los proyectos rara vez tienen más de dos miembros y que hay muchos proyectos repartidos en el equipo cuyos desarrollos se van entremezclando, hace algo complicado poder implantar un enfoque Scrum “puro”.

Por otro lado, algunos proyectos tienen varios equipos como actores implicados. En especial, hay un proyecto en el que hay dos miembros de distintos equipos desarrollando de forma esporádica y otro equipo que hace labores de gestión comercial. Como los papeles no son permanentes, las tareas pueden cambiar de propietario sin previo aviso. Esto hace muy complicado la visualización del proyecto en su conjunto, por lo que sólo podremos gestionar las tareas que nos asignen a nuestro equipo.

5.1.2.6 Kanban no obliga a olvidar el resto de prácticas ágiles

Kanban es sólo un método de planificación de las tareas, pero no aborda aspectos como buenas prácticas de desarrollo, calidad software, tests o estudios de usabilidad.

Esto lo hace perfecto para nuestro equipo, ya que mientras Kanban ayuda a una planificación y control de tareas eficiente, por otro lado, podemos realizar un proceso de mejora de desarrollo software paralelo.

5.2 Autoorganización

Desde el punto de vista de los marcos de trabajo ágiles, los equipos que mejor trabajan son los que se organizan ellos mismos.

Pero en este caso, dado que cada uno de los desarrolladores trabaja en su propio proyecto (salvo casos puntuales), por lo que no habrá que organizar a varios desarrolladores. En cambio, habrá que organizar el tiempo de cada uno de los desarrolladores individualmente para que su rendimiento sea

óptimo.

Para evitar la existencia de desarrolladores “héroes” con conocimiento único de un proyecto, se irá compartiendo poco a poco el desarrollo entre varios de los miembros del equipo. Así, hasta que el código pase de ser “propiedad común” de todo el equipo.

5.3 El foco de un desarrollador ha de ser el desarrollo

En mi lugar de trabajo se pierde mucho tiempo por interrupciones, bien en forma de ruido, o cuando acude un compañero a preguntarnos una duda. Mi objetivo es acabar con esto.

Para eso, mi objetivo es actuar como facilitador. ¿Qué es un facilitador? Un miembro del equipo que les dé apoyo para cualquier necesidad que tengan los desarrolladores y que les evite las tareas más pesadas (normalmente la comunicación con dirección o clientes).

Algunas veces los desarrolladores tendrán que comunicarse con los clientes (vía correo electrónico) cuando hayan terminado una tarea, pero el grueso de la comunicación estará en mis manos. Además, tendrá la misión de orientar y revisar esa comunicación.

También, deberá velar porque mi equipo tenga trabajo. Muchas veces hay “tiempo muertos” entre proyectos debidos a que hemos de esperar a que el cliente apruebe una mejora y deberá gestionar que el equipo tenga carga de trabajo. Usaremos esos tiempos de espera para formación o refactorización de código.

Por otro lado, también me encargo de distribuir el trabajo. Dado que es un equipo heterogéneo, también he de conocer las habilidades de cada miembro de mi equipo para saber si puede afrontar un desarrollo de un proyecto o funcionalidad determinada.

También libero a mi equipo de la planificación, en el sentido de que yo soy el que controla los tiempos de desarrollo y las entregas que se han de realizar a clientes.

Por último, soy el último responsable de la calidad del software, lo que me obliga a revisar la funcionalidad desarrollada por cada uno de los desarrolladores de mi equipo.

Como se puede ver, actúo más que facilitador sólo y en exclusiva, como apoyo de mi equipo, como si fuera un entrenador que vela por la integridad de mi equipo y la calidad de su trabajo.

5.4 Código de calidad

Actualmente los desarrolladores no se preocupan por hacer código de calidad. Creo firmemente que el código ha de ser fácilmente mantenible. Un código sin comentarios o con un diseño demasiado complejo, por ejemplo dificultan el mantenimiento, enlenteciendo la corrección de un fallo o la adición de funcionalidad extra.

No es una metodología ágil, pero quiero combinar la Programación Literaria² con el desarrollo de

² La [Programación Literaria](#) [KNUTH84] es una técnica desarrollada por Donald Knuth en la que el desarrollador redactaba el comportamiento de cada una de las funciones del código en el mismo código. Luego un procesador separaba el código de los comentarios, generando el código ejecutable y la documentación. Nosotros nos

código limpio, de manera que el código describa en los comentarios su funcionalidad y tenga una estructura adecuada y fácil de entender.

Hay ciertas escuelas de desarrollo que están en contra del uso de comentarios en el código, pero dado que muchas veces tenemos proyectos sin ninguna documentación, creo que lo mínimo es que el código esté comentado. Eso sí, los comentarios deberán ser útiles e ir al meollo del asunto, explicando los motivos de cada sentencia comentada.

A continuación describimos una serie de prácticas que son completamente necesarias para que un equipo desarrollo software de calidad y que serán las que promovamos en nuestro equipo:

5.4.1 Minimización del estado global

Cualquier desarrollador que se precie sabe que el estado global es el enemigo en cuanto al desarrollo de software. Dado un software con una alta deuda técnica, evitar transiciones a estados inconsistentes y los efectos colaterales de los nuevos cambios se convierte en una tarea de gran esfuerzo.

Algunos autores como [MOSELEY06] han propuesto nuevos paradigmas o herramientas de desarrollo para minimizar este problema.

También, el paradigma de programación funcional busca trasladar el concepto de función matemática a la programación, de manera que el resultado siempre dependa sólo y exclusivamente de los parámetros.

Lamentablemente, no ha habido una gran popularización de estos enfoques, por lo que la calidad del código sigue dependiendo mucho de la calidad que le quiera dar el equipo de desarrollo.

En este caso concreto, promoveré las siguientes acciones:

- Desechar completamente los objetos y variables globales.
- Los métodos podrán consultar o mutar el estado del objeto, pero no ambas cosas.
- Hacer que las funciones (y métodos) tengan el mínimo posible de efectos colaterales (o el mínimo posible de cambios de los atributos de objeto).

5.4.2 Orientación a objetos

La orientación a objetos (bien implementada) es un arma muy potente para estructurar el código. Deberemos tratar de definir clases con una única misión y con una buena ocultación de su parte privada.

Seguiremos también el principio de los *Naked Objects* [PAWSON04], de manera que, al menos, toda la lógica esté encapsulada en los objetos y no en las vistas, tal y como se ha estado haciendo hasta ahora.

quedaremos en hacer comentarios de calidad que expliquen el flujo de ejecución y el contexto donde se usa cada módulo. De esta forma, lograremos mejorar el mantenimiento y desarrollo de nuevas funcionalidades.

5.4.3 Principio de única responsabilidad

De acuerdo a Robert C. Martin [MARTIN08], este principio es el siguiente: “una clase sólo debería tener un motivo para cambiar”. Este autor se refiere a que cada clase (o módulo) sólo debería tener una responsabilidad sobre una parte del conjunto total del software, de manera que todas sus operaciones estuvieran en esa línea.

En el código heredado de otros proyectos nos hemos encontrado grandes ficheros de código en los que una misma clase actúa de múltiples formas. Esto complica de forma inmensa el desarrollo de software y procederemos a segmentar las clases en otras que tengan menor grado de responsabilidad.

5.4.4 Nombres de variables

Tanto Steve McConnel en [MCCON04] como Robert C. Martin en [MARTIN08], hacen hincapié en la nomenclatura de variables de forma insistente. De hecho, Martin esgrime que un código con una buena estructura y unos buenos nombres de variables no necesita comentarios apenes.

Algunos miembros menos experimentados del equipo no están acostumbrados a estas prácticas y sólo formación con buen material como las referencias anteriormente comentadas, una tutorización permanente y revisiones de código harán que esto mejore.

5.4.5 Reducción de la complejidad ciclomática

La complejidad ciclomática es una medida de calidad de código que indica las distintas bifurcaciones que tiene un módulo de código. También permite ver lo sencillo que es de entender un código ya que a mayor valor de complejidad ciclomática, más complejo es éste. Por eso, tener como objetivo tener una complejidad ciclomática mínima en cada método o función es fundamental.

Si seguimos el principio de única responsabilidad y tratamos de tener módulos de pequeño tamaño, esta buena práctica casi que aparecerá sola en nuestro equipo de trabajo.

5.4.6 Refactorización

La mejora continua del software es importante y para ello voy a impulsar el aprendizaje de los patrones de refactorización más comunes [FOWL99].

Es cierto que es complejo implantar refactorización en entornos que no usan pruebas, como es el nuestro, pero trataremos de dar cierta cobertura de pruebas a nuestros proyectos y refactorizar aquellas partes que puedan probarse de forma sencilla.

5.4.7 Revisiones de código

El código ha de ser revisado por aquellos miembros con más experiencia, de manera que puedan evaluar si, efectivamente, el desarrollador autor del código está siguiendo las buenas prácticas o no. En el caso de que no, será el encargado de formarle para que las siga.

En Thoughtworks hacen revisiones de código de equipos dispersos por el mundo mediante el análisis de código de las revisiones de los desarrolladores [DISHMAN15]. Dado que nosotros

estamos todos en la misma oficina, podremos revisar el código *in situ*.

5.4.8 Programación por parejas

No vamos a implantar Programación Extrema [BECK04], pero para determinadas partes de algunos proyectos sí que vamos a trabajar a la vez sobre el mismo problema. Está comprobado que es una buena forma de abordar problemas que no son triviales, además de que mejora el trabajo en equipo y la dispersión de conocimiento.

5.5 Gestión del tiempo

Como comenté más arriba, hay muchas interrupciones. Por eso, estoy promoviendo el uso del método Pomodoro para la gestión del tiempo.

Mi objetivo es reducir las distracciones y conseguir periodos de concentración en los que la atención en el desarrollo de cada miembro del equipo sea máxima.

5.6 Entregas frecuentes

Actualmente, en los desarrollos nuevos, sólo se entrega al cliente al finalizar el proyecto. Es cierto que se tiene el *storyboard* validado y que en la mayoría de los se adapta bastante bien a lo que quería el cliente, pero esta forma de trabajar decrementa la velocidad de los proyectos y hace que éstos se dilaten en el tiempo.

Mi objetivo es que cada semana entreguen un subconjunto de las tareas que están desarrollando al cliente.

Para cada proyecto tenemos la intención de crear servidores de prueba de forma que se le pueda enseñar siempre cada viernes al menos una tarea que hemos desarrollado al completo.

Las entregas frecuentes por un lado previenen los problemas con requisitos cambiantes y por otro, refuerzan la relación con el cliente, que ve cómo se genera valor cada semana. Eso refuerza su confianza con el equipo de desarrollo, favoreciendo su relación con nosotros y mejorando su disposición a trabajar con nosotros para sacar adelante el proyecto.

Es complicado establecer un día de entrega determinado, pero trataré que todos los miembros de mi equipo realicen entregas cada semana, de manera que el cliente vea que su proyecto avanza.

5.7 Colaboración con el cliente

Dado que en muchos proyectos no se hacían entregas hasta que el proyecto estaba prácticamente terminado, es lógico pensar que no habían ningún tipo de colaboración con el cliente. De hecho, se veía al cliente como un “enemigo” cuyo objetivo era “retrasar el proyecto al máximo”, ampliándolo de forma ininterrumpida hasta que tuviese más funcionalidad que la que había pagado.

Esta imagen de cliente “enemigo” es muy perjudicial, ya que hace que trabajemos contra él en vez de con él.

Voy a incentivar que los desarrolladores entiendan las necesidades del cliente y no sólo se hagan eco de lo que les dice³, sino que establezcan conversaciones en las que se puedan sacar sus necesidades reales.

El objetivo es no ser meros codificadores, sino poder buscar soluciones lo más efectivas posibles a las necesidades del cliente.

5.8 Documentación

La documentación suele ser infravalorada por los desarrolladores. La mayoría de los casos se considera un “estorbo” ya que no es completamente necesaria para tener un software que funcione. Pero sí que es completamente necesaria para proporcionar los conocimientos base para ampliar y mantener un proyecto.

Vamos a usar el enfoque de diagramas C4 [BROWN14] junto con documentos redactados en texto plano como documentación de los proyectos (más allá de los comentarios en el código). Los diagramas C4 proporcionan una panorámica del proyecto a distintos niveles, y para la definición de flujos de ejecución usaremos o diagramas de secuencia o texto plano.

Lo importante no es si los diagramas son UML o son de otro tipo, lo importante es que la documentación sea sencilla, fácil de leer y refleje el estado real y actual del proyecto.

5.9 Pruebas e integración continua

Quiero implantar un proceso de pruebas para cada uno de los módulos de nuestros desarrollos. Además, del uso de integración continua para evitar subir software con fallos a los servidores de producción.

Espero comenzar con TDD (*Test Driven Development*, Desarrollo Dirigido por Tests) y luego poder usar esas pruebas como parte de las pruebas de integración.

Comenzaremos probando el software de servidor con tests de unidad y luego el software de cliente (código *Javascript*) con tests funcionales.

5.10 Dar valor al software

Cuando llegué al equipo, me encontré con proyectos en los que el cliente no tenía voz ni voto en lo que se desarrollaba. Parece un contrasentido, pero era el desarrollador el que, de alguna forma, iba escogiendo las tareas según éste veía adecuado.

En un proyecto concreto, el problema es que un desarrollador nunca llegaba a terminar las tareas completamente, por lo que este proyecto ha estado 6 meses sin entregar software funcional⁴ al cliente.

³ En mi experiencia, lo que dice el cliente y lo que necesita son dos cosas muy distintas. Hasta que no entendemos su negocio tan bien como él, no podemos darle una solución para sus verdaderos problemas.

⁴ El desarrollador también hacía mejoras para otros productos software del cliente, por lo que de alguna forma sí le estábamos dando software funcional, sólo que exclusivamente en uno de sus proyectos (no en el otro).

Esto no puede suceder más. Se ha de desarrollar de forma incremental e iterativa, de manera que el cliente pueda decidir hacia dónde va el desarrollo software con acceso a un prototipo software funcional.

5.11 Otras buenas prácticas

Si bien no son prácticas ágiles, considero que es necesario promover estas buenas prácticas en el equipo:

5.11.1 Formación continua

Un desarrollador que no se forma se queda atrás. Les he implantado un plan de formación y para empezar les he animado a apuntarse a un curso de metodologías ágiles gratuito por internet y el siguiente paso será enseñarles los principios del *código limpio*. Después continuaremos con las pruebas, de manera que pasemos de un enfoque sin pruebas a un enfoque TDD.

Mi objetivo es hacer de la mejora continua la base fundamental de su trabajo, buscando siempre ser mejores cada día, de manera que progresen y no se queden estancados.

5.11.2 Autoevaluación

Otra forma de potenciar que los desarrolladores mejoren es la autoevaluación de su rendimiento cada semana. Para ello, les he animado a sumar las horas de desarrollo efectivo, registrarlas cada semana y así poder ver la evolución de cada uno de los desarrolladores a lo largo del tiempo.

El objetivo es siempre evitar las interrupciones y lograr velocidades de desarrollo de aproximadamente el 70% del tiempo que estamos en la oficina, lo que según los estándares de la industria se considera aceptable.

5.11.3 Potenciar estudios de usabilidad

Actualmente no se hace ningún estudio de usabilidad sobre las aplicaciones más allá del *storyboard* inicial. Este *storyboard* ayuda al cliente a validar la funcionalidad y ahí termina la implicación del cliente.

Creo que lo primero que hay que hacer es promover el conocimiento de las prácticas heurísticas de usabilidad entre el equipo y su aplicación en todos los proyectos.

También, motivar a que se haga uso siempre de ayudas adicionales en las conversaciones con el cliente, como bocetos de interfaces, *storyboards* y diagramas.

Dado que el software es inmaterial y que trabajamos con ideas, las herramientas visuales son las que hacen que el cliente comprenda a la totalidad la idea que le estamos proponiendo.

6 Metodología de trabajo

6.1 Introducción

Como se ha indicado antes, en la sección de “Prácticas ágiles seleccionadas”, es complicado usar Scrum dada la forma de trabajar en los proyectos y la estructura del equipo. Por eso se ha optado por tomar como marco de trabajo básico Kanban.

Pasamos a describir cada uno de los elementos de esta técnica y la personalización que hacemos para que se adapte mejor al equipo.

6.2 Estructura del tablero Kanban

6.2.1 Estados de las tareas

En Kanban los estados de las tareas se modelan como columnas del tablero, por lo que para todos los tableros de los proyectos, tenemos los siguientes estados:

- **Pendientes de estimación**^{*}: mejoras que deben ser estimados por el equipo técnico.
- **Estimadas**^{*}: las tareas han sido estimadas (en un rango de horas). El equipo comercial
- **Pendientes de confirmación por cliente**^{*}: las tareas que están en esta columna han de ser aprobadas por el cliente. Normalmente esto ocurre cuando son mejoras específicas solicitadas por un cliente pero que se aplicarán a toda la plataforma.
- **Pendiente de desarrollo**: indica que la tarea está en cola para entrar a desarrollo. El orden de las tareas (de arriba a abajo) indica la prioridad. El equipo comercial puede cambiar la prioridad de estas tareas en este estado según vea las necesidades del cliente.
- **En desarrollo**: las tareas que se están desarrollando.
- **En revisión técnica**: las tareas que están completadas pero deben pasar una revisión. Normalmente las revisiones son revisiones de calidad de software y de funcionalidad por otro miembro del equipo.
- **Pendiente de subida a producción**: se ha solicitado la actualización de los servidores a administración de sistemas. En cuanto se haya subido la revisión con cada una de las tareas en esta columna, éstas irán pasando al siguiente estado.
- **En revisión comercial**: el equipo comercial revisa las tareas por si hay algún aspecto que no se ha implementado o por si necesitan formación de cara a explicar al cliente cómo usar esa funcionalidad.
- **Comunicado al cliente**^{*}: se ha informado al cliente de que el cambio en el software ha sido realizado y su sitio web está actualizado.

* Estas columnas sólo están presentes en tableros compartidos con el equipo comercial.

- **Terminado:** las tareas están terminadas y todos los equipos saben que han sido desarrolladas.

Un cliente concreto que colabora mucho en el proyecto es directamente el encargado de dar prioridades y crear las tareas con las incidencias o cambios de su sistema, pero por lo general somos el equipo técnico y el equipo comercial los encargados de gestionar los tableros.

6.2.2 Límite de *Work in Progress* (WIP)

Al principio no implantamos un máximo número de tareas por columna, pero vimos que era necesario para evitar los cambios de contexto.

Así, implantamos un máximo de WIP en el estado de “En desarrollo” de 2. De esta forma no se recomienda que un desarrollador esté con más de dos tareas simultáneamente.

El motivo por el que escogemos este límite se debe a que los proyectos son todos prácticamente de un único desarrollador en un momento dado. Esto hace que un límite razonable sea el de 2.

Normalmente, las tareas son resueltas por un único desarrollador, así que en el caso de proyectos de varios desarrolladores, incrementamos el WIP en 2 por cada desarrollador adicional que no trabaje junto con otro desarrollador (de forma similar a como se hace en la *Extreme Programming*).

Esta limitación no tenemos forma de obligarla desde la herramienta que usamos, pero como promovemos la interiorización y la disciplina para de este tipo de limitaciones por lo que no hace falta una limitación externa.

6.3 Herramientas software

6.3.1 Trello

Para implantar Kanban vamos a hacer un tablero de Trello [TRELLO] para cada proyecto.

Hemos hablado de que en función de la tipología de proyecto habrá unas columnas u otras. Básicamente, si intercede el equipo comercial deberemos incluir unos estados u otros para que ellos puedan también usarlo como herramienta para sus procesos de trabajo.

Por ejemplo, se muestra a continuación un tablero con las tareas de un proyecto que está en desarrollo y en el que participa el equipo comercial:

This screenshot shows a Trello board titled "Tareas de Proyecto". It features six columns: "Pendiente de presupuesto por equipo técnico", "Pendiente de confirmación por cliente y equipo comercial", "Pendiente de desarrollo", "En desarrollo por equipo técnico", "Pendiente de subida", and "En revisión por equipo comercial". Each column contains several cards with task details, such as "Revisar presupuesto de desarrollo", "Revisar documentación de diseño", and "Realizar diseño de la interfaz de usuario". A "Visible para el equipo" button is present at the top left.

No se muestran las columnas “Comunicado al cliente” y “Terminado” por limitaciones de visualización de la aplicación web a la hora de realizar la captura en imagen.

A continuación se muestra un tablero Kanban de un proyecto de desarrollo puro⁵ en el que el equipo comercial no tiene ningún papel:

This screenshot shows a Trello board titled "Desarrollo de Proyecto". It features four columns: "Pendiente", "En proceso", "Pruebas", and "Hecho". The "Hecho" column is expanded to show a list of completed tasks, including "Realizar diseño de la interfaz de usuario", "Revisar documentación de diseño", and "Realizar diseño de la interfaz de usuario". A "Visible para el equipo" button is present at the top left.

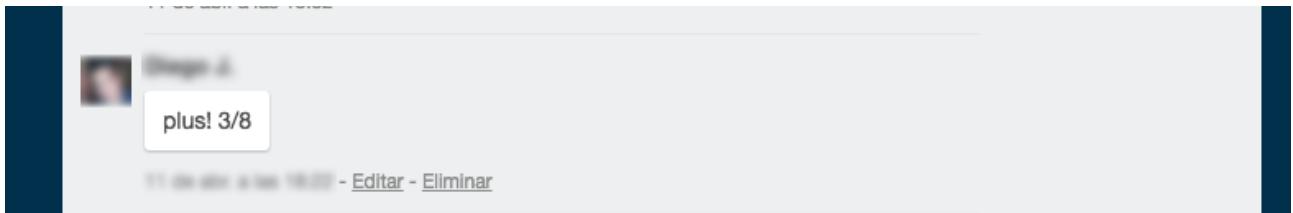
⁵ El cliente en este proyecto nos ha entregado un análisis basado en un *storyboard* para usarlo como base del diseño software y por supuesto, del desarrollo como producto.

6.3.2 Plus for Trello

Además del propio tablero, vamos a usar un complemento de Google Chrome que nos dará estadísticas sobre la eficacia de los tiempos de estimación, indicando la diferencia entre la estimación y el tiempo consumido por tarea. Esta extensión es “Plus for Trello”.

Plus for Trello permite la inclusión de los tiempos estimados y consumidos en cada tarea en cada propia tarjeta. Lo hace introduciendo comentarios con un formato particular.

Mostramos a continuación un comentario realizado sobre una tarea que indica que se han gastado 3 horas de las 8 en las que se estimó que se iba a realizar:



Una tarjeta puede tener muchos comentarios con distintos valores. Plus for Trello asume que son mediciones incrementales, por lo que habrá que sumar tanto su tiempo consumido como el tiempo estimado.

Estos comentarios podrán ser leídos por PyStats-Trello [PYSTTRLO], generador de gráficas y estadísticas para Trello desarrollado por mí (y de código abierto). De esta sencilla forma, tendremos todos los datos centralizados, desde los tiempos de espera hasta los tiempos efectivos de desarrollo y las estimaciones hechas por los desarrolladores.

6.4 Prácticas ágiles no canónicas de Kanban

Si bien no creo que se pueda implantar directamente Scrum en un equipo que ha estado usando un enfoque no-ágil durante tanto tiempo⁶, sí que creo que se pueden implantar algunas partes de este marco de trabajo. En nuestro caso dos elementos:

6.4.1 Reuniones diarias

Las reuniones diarias se hacen con la intención de que todos los desarrolladores vayan poco a poco aprendiendo del resto de los proyectos. De esta forma, en dicha reunión, cada uno de los desarrolladores ha de indicar las tareas que va a hacer en el día de hoy y las tareas que terminó ayer. Además, si ha aprendido algo reseñable para el resto del equipo lo comparte.

6.4.2 Reuniones de retrospectiva

Cada viernes a última hora (14:30) tenemos media hora de reunión en la que compartimos “cómo ha ido la semana”. Esta reunión es un punto de encuentro entre los desarrolladores, de manera que los

⁶ Necesitamos primero ir poco a poco, educando a los clientes e incluso a nosotros mismos en el marco de trabajo ágil antes de pasar a usar algo como Scrum. Sobre todo porque nadie tiene experiencia en mi empresa en esto. El objetivo de este proceso es dar un primer paso en metodologías ágiles para, llegado el caso, plantearnos usar Scrum en algún proyecto concreto.

objetivos son entre otros:

- Compartir lo que se ha aprendido esa semana.
- Hacer una presentación sobre un proyecto concreto.
- Proponer mejoras sobre el marco de trabajo.
- Solicitar ayuda o asistencia sobre cualquier tema del ambiente de trabajo.

Estas reuniones contribuyen a generar sentimiento de pertenencia al equipo. Un equipo en el que no haya comunicación es un “equipo muerto”.

6.4.3 Colaboración y reuniones con dueños del producto

Un aspecto fundamental es la colaboración con los dueños de producto. Los desarrolladores no podemos desviarnos de las necesidades del dueño del producto y además, de esta forma evitamos diferencias muy acusadas entre sus necesidades y el producto software.

Así, bien sean estos el equipo comercial o un cliente externo, hemos definido un protocolo de reuniones para intercambiar información y evaluar el estado del proyecto. Estas reuniones se celebrarán con el equipo comercial de cada proyecto con periodicidad semanal y en el caso de un proyecto cuyo cliente es externo, se harán de forma mensual.

Eso sí, es cierto que no todos los proyectos permiten este tipo de colaboración. Bien por falta de interés del dueño de producto, por falta de conocimiento o por proyectos heredados con un presupuesto cerrado.

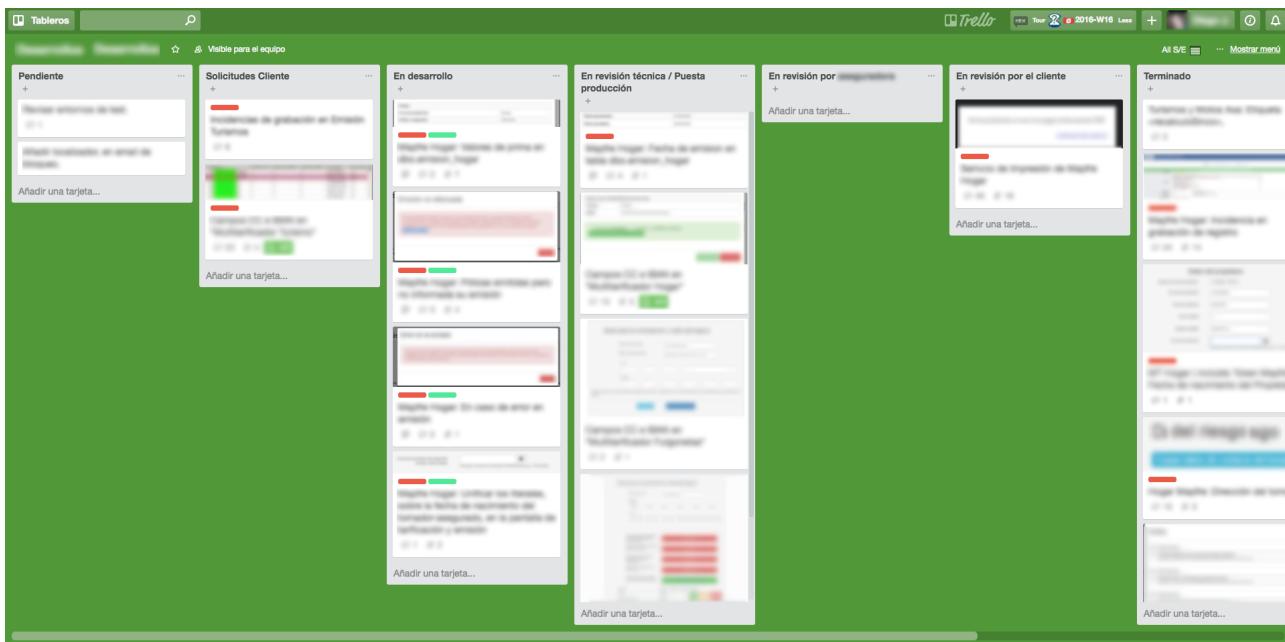
En este caso, compartimos un informe del estado del proyecto (normalmente en forma de hoja de cálculo). En este informe se indican las tareas que estamos desarrollando, el coste de cada una de ellas y si han de revisarlas o no. Además, este informe nos sirve como registro de la evolución del proyecto.

A continuación mostramos un registro con las acciones realizadas de uno de nuestros proyectos:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Semana	Concepto	Origen	Tiempo	Importe	Revisado	Cómo se validó	Fecha validación	Fecha facturación	Importe total	Importe facturado	Resto a facturar	Precio hc
2						Si							
3						Si							
4						Si							
5						Si							
6						Si							
7						Si							
8						Si							
9													

También, le hemos propuesto a los clientes más avezados el uso de tablero Kanban (de forma adicional a la hoja de costes).

Mostramos a continuación uno de estos tableros en los que hay relación con el cliente. Una de las columnas es “Solicitudes de cliente” y ahí se puede ver cómo el cliente ha incluido dos incidencias críticas (etiquetas con color rojo):



El permitir este nivel de colaboración tiene sus problemas, claro está. En esta captura, de hecho, vemos cómo en el estado “En desarrollo” hay 4 tareas, superando el WIP máximo de 2⁷. Esas tareas

⁷ Recordemos que para que Kanban funcione correctamente ha de haber un límite en el número de tareas que se están desarrollando a la vez. Aunque este tablero no lo indica explícitamente, hemos puesto un límite de 2 tareas en desarrollo como máximo. Hay que tener en cuenta que ese límite es sólo para los proyectos en los que hay un único

las ha creado el cliente *motu proprio* y deberemos moverlas a “Solicitudes cliente” y poco a poco promover una cultura de colaboración en la que se siga un pequeño protocolo.

6.4.4 Colaboraciones con otros desarrolladores

Es vital que haya comunicación con otros desarrolladores. Bien del mismo equipo o de otros. El conocimiento obtenido con base en la experiencia debe distribuirse entre los desarrolladores y la mejor forma de que esto suceda es mediante la comunicación.

No sólo comunicación verbal, sino de cualquier forma. Muchas veces se aprende más por imitación, al ver a un compañero resolver un problema, que intentando resolver por medios propios el problema (además de que es mucho más rápido).

Eso sí, la comunicación deberá servir para algo. El objetivo es crecer como desarrolladores, no perder el tiempo en ocio o conversaciones que no llegan a nada.

7 Aplicación de los estudios de usabilidad al proceso de desarrollo

7.1 Introducción

En la selección de prácticas ágiles hemos indicado la tremenda necesidad de aplicar buenas prácticas de usabilidad debido al gran número de incidencias que generan los clientes por no saber usar nuestras aplicaciones web.

Esto ocasiona la creación de un gran número de incidencias en las que los clientes no son capaces de trabajar con la aplicación. Según mi experiencia, esto se debe a varios motivos:

1. Existencia en el sistema de un flujo de trabajo distinto al de su negocio.
2. Incapacidad para reconocer las funcionalidades en el sistema.
 1. Porque las funciones no son evidentes en el sistema.
 2. Las funciones no se adaptan completamente a la forma de trabajar del cliente en su negocio.
3. Olvido de cómo usar la funcionalidad del sistema.

Creo que una buena forma de resolver las incidencias creadas por situaciones 2, 2.1, 2.2 y 3 es la aplicación de buenas prácticas de desarrollo de software usable.

7.2 Formación

Dado que ninguno de los desarrolladores tiene ningún conocimiento concreto sobre usabilidad. Vamos a seguir un MOOC de las *Open Universities Australia* que contiene unos principios básicos de experiencia de usuario [WEBUX]. El último módulo de este curso masivo trata especialmente de usabilidad, por lo que nos centraremos en éste.

También les he recomendado la lectura de las referencias bibliográficas *Don't Make Me Think* [KRUG14] y *Usability Engineering* [NIELSEN93].

Lamentablemente ni en el Grado de Ingeniería Informática ni en los ciclos formativos de grado superior se imparte formación sobre esta disciplina, por lo que no podremos profundizar tanto como me gustaría.

Así que este proceso de formación del equipo le dará una base sobre la que seguir trabajando.

7.3 Modelo iterativo de usabilidad

El objetivo es poder ir mostrando distintos bocetos e interfaces a los distintos miembros del equipo que no participan en el proyecto y recibir de ellos retroalimentación.

Obviamente lo óptimo sería contar con usuarios con menos conocimientos que un desarrollador,

pero dado que esto no es posible, tendremos que adaptarnos haciendo de usuarios nosotros. Esto no supondrá un problema ya que, al estar fuera del proyecto, este desarrollador no tendrá conocimiento sobre el funcionamiento del proyecto y por tanto, actuará de forma similar a como lo haría un usuario “normal”.

Decimos que es un modelo iterativo, porque, aunque tiene 4 pasos, el cliente y los desarrolladores irán refinando las interfaces poco a poco hasta obtener el resultado deseado.

A continuación, mostramos el proceso de 4 pasos que han de completar todas las interfaces que desarrollemos de aquí en adelante:

7.3.1 Bocetos

Los bocetos son representaciones simples de interfaces de usuario.

Son una herramienta básica a la hora de poder mostrar interfaces rápidamente a los clientes y a otros usuarios y serán los primeros elementos que se evalúen y se refinen de forma iterativa.

Los hará el mismo desarrollador para poder comunicar cuál será el diseño final tanto al resto de miembros del equipo, como al cliente o a los diseñadores.

Si el sistema es complejo, se harán varios y se dispondrán en forma de *storyboard*.

7.3.2 Evaluaciones heurísticas

Una vez completado el boceto de la interfaz, un miembro del equipo será el encargado de hacer una evaluación heurística de la interfaz.

Las evaluaciones heurísticas tienen como objetivo aplicar las buenas prácticas de usabilidad a un diseño de una interfaz. De esta manera, se detectan los errores de forma temprana y el resto de las pruebas pueden consumir menos tiempo.

En principio las evaluaciones heurísticas las haré yo hasta que el resto del equipo tenga suficiente experiencia en cuanto a los principios de usabilidad [NIELSEN95] [UBOK].

7.3.3 Búsqueda de soluciones

Muchas veces, los problemas a los que nos enfrentamos ya están resuelto, y el mundo de la usabilidad no va a ser menos.

En esta fase vamos a intentar buscar diseños que solucionen los problemas que no sabemos solucionar o para los que no tenemos el suficiente tiempo. La búsqueda de este tipo de “patrones de interacción” se hará tanto entre nuestros proyectos como en la red, siempre buscando un patrón que haya sido un caso de éxito.

Por ejemplo, mostramos una interfaz que diseñamos para un proyecto y que dada su aceptación entre los usuarios de dicho proyecto la hemos usado como patrón en otros. Esta interfaz es un buscador avanzado que permite realizar consultas complejas sobre usuarios, productos u otras entidades. Se puede ver cómo se pueden definir distintos criterios de búsqueda. Éstos se estructuran

como una disyunción de “opciones de consulta” que son conjunciones de condiciones, de manera que internamente la condición de selección es una forma normal disyuntiva [DNF]:

Criterios de búsqueda

The screenshot shows a search interface with two main sections: 'Criterios de búsqueda' (Search Criteria) and 'Resultados' (Results).

Criterios de búsqueda: This section contains two rows of search fields. The first row includes dropdown menus for 'Operador' (Operator), 'Campo' (Field), and 'Valor' (Value), with buttons for 'Igual a' (Equal to) and 'Quitar' (Delete). The second row includes dropdown menus for 'Operador', 'Campo', and 'Valor', with buttons for 'Igual a', 'Servicio 1', 'Quitar', and 'Añadir búsqueda en campo' (Add search in field). Below these rows are buttons for 'Quitar esta opción de consulta' (Remove this search option) and 'Añadir nueva opción de búsqueda' (Add new search option).

Resultados: This section displays a table of search results. The columns are labeled: 'ID', 'Nombre', 'Apellido', 'Sexo', 'Número', 'Número 2', 'Número 3', 'Población', and 'Acciones'. The table lists 10 entries, all from 'Granada (Granada)' with 'Ver ficha' (View file) under 'Acciones'. Navigation buttons at the bottom left include arrows for page navigation and a 'Ver todos' (View all) link.

Hay mucha información en la red sobre este tipo de patrones. Considero que [WELIE] [YAHOODPL] y [ERICKSON04] son lo suficientemente completos para usarlos como referencia en nuestros desarrollos.

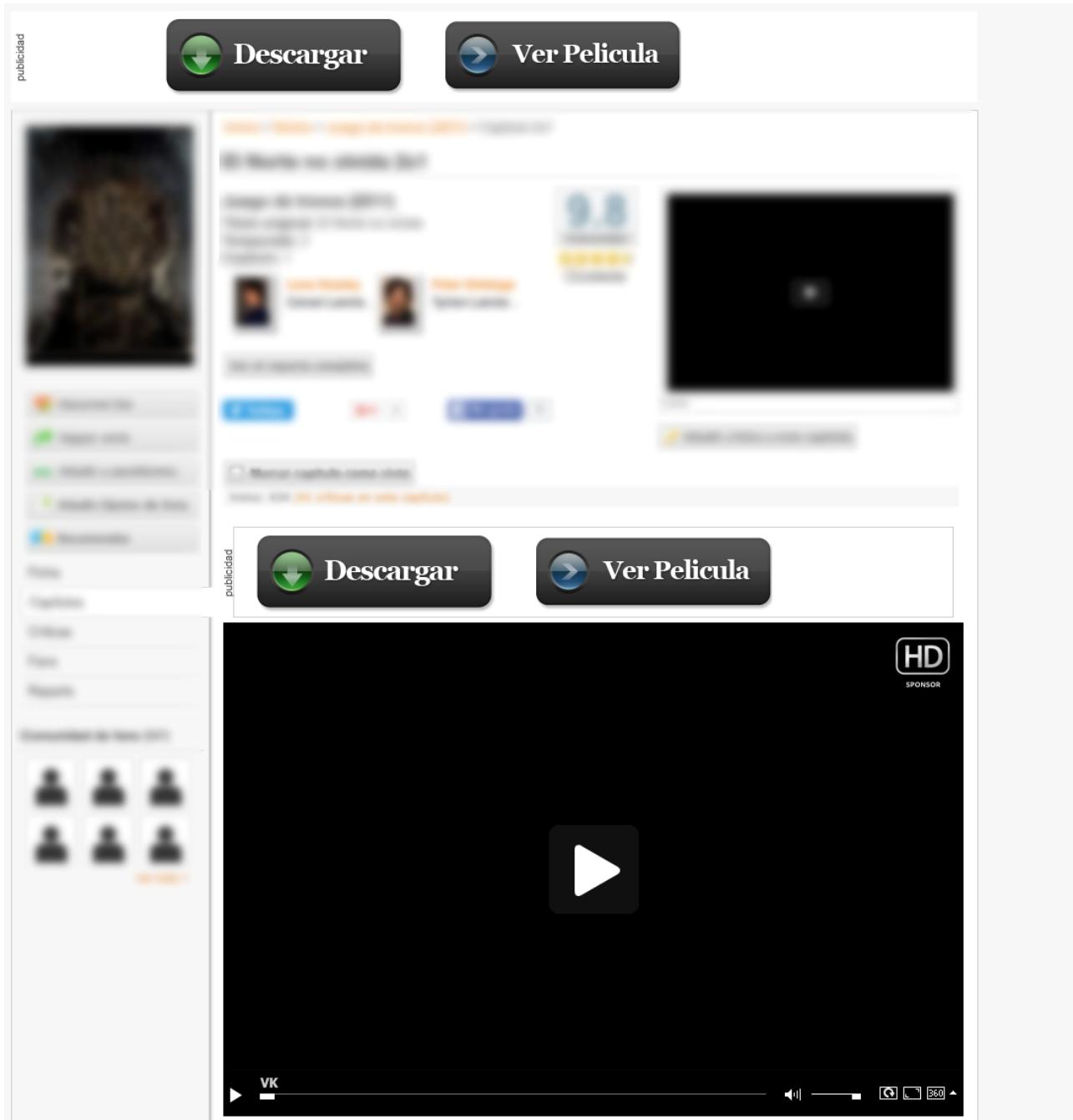
7.3.4 Evitar implementación de “Dark patterns”

En nuestra industria se conoce como “Dark Pattern” [DARKPATT] a determinados patrones de interacción que tienen como objetivo principal engañar al usuario. Normalmente son técnicas que hacen que los usuarios pulsen en botones que no hacen lo que indican que han de hacer o inclusión de capas transparentes sobre páginas web de manera que se pueda capturar la acción de pulsar sobre una zona y abrir otra ventana con publicidad, etc.

Este tipo de prácticas son comunes en las páginas de dudosa legalidad (compartición de ficheros o películas con derechos de autor...). Pero en el mundo publicitario se están haciendo cada vez más frecuentes⁸.

Por ejemplo, aquí vemos un famoso portal de descarga y visualización de películas y series que abre ventanas cuando se pulsan los elementos que no están difuminados:

⁸ De hecho, conozco a un programador de Javascript cuyo cometido es el de crear este tipo de interacciones en páginas de los clientes de la empresa para la que trabaja.



Hasta ahora no ha habido ningún cliente que nos haya pedido de forma explícita el uso de “dark patterns”. Pero hemos de anticiparnos a ese tipo de solicitudes y disuadir a los clientes de su uso por los siguientes motivos:

- **Ética del ingeniero y desarrollador:** como ingenieros y desarrolladores debemos hacer avanzar a la sociedad, no debemos sólo preocuparnos de nuestro salario y de las órdenes que recibamos.
- **Disminución de la confianza del usuario en el servicio y consiguiente pérdida de**

usuarios: si el usuario ve que las acciones no indican lo que realmente hacen, dejará de visitar la página y buscará una que sí sea veraz.

- **Reputación de la empresa:** ¿qué reputación tendrá una empresa que acepta e implementa este tipo de prácticas? ¿Merece la pena?

7.3.5 Tests de usabilidad

Para interfaces muy complejas, también haremos uso de tests de usabilidad. No los haremos con usuarios externos ni serán muy exhaustivos, pero nos servirán para saber si alguna tarea o escenario es muy complejo de resolver para un usuario medio.

Lo primero de todo es que el sujeto del test será otro de los miembros del equipo. Eso sí, no podrá haber participado en el desarrollo ni conocer el proyecto. Según mi experiencia, las pruebas (tanto software como de usabilidad) no se realizan en profundidad si el que las hace es uno de los desarrolladores de la aplicación.

Es evidente, que este test requiere que la aplicación esté funcionando. No requiere que la aplicación esté completa (pueden ser datos de prueba), pero tiene que dar la misma respuesta que daría en el sistema real. Si no, el test no tiene sentido.

Así, uno de los desarrolladores de la aplicación planteará un escenario al usuario. Ese escenario ha de tener los siguientes datos:

- Objetivos del usuario. Por ejemplo: “comprar una entrada”.
- Escenario del usuario. Por ejemplo: “se supone que has iniciado sesión con un usuario con permisos adecuados”.

Uno de los desarrolladores de la aplicación tomará entonces notas sobre el comportamiento del usuario:

- Tiempo aproximado que tarda en hacer cada una de las tareas.
- Comentarios y preguntas que hace el usuario.
- Problemas que detecta el usuario.
- Pensamientos en voz alta y otros comentarios.

No es un test completamente ortodoxo ya que no se hace con usuarios reales y si asumimos que los desarrolladores son “usuarios avanzados” peor aún, pero es lo mínimo que se puede hacer.

Por último, este “test” más que encontrar fallos de usabilidad de forma empírica creará una conversación entre desarrolladores que mejorará la usabilidad al compartir experiencias y conocimientos entre los distintos desarrolladores.

7.3.6 Accesibilidad

Consideramos que toda web que hagamos ha de ser accesible, por lo que seguiremos las guías Web

Content Accessibility Guidelines [WCAG20] del W3C.

Si bien es cierto que, hasta ahora, ningún cliente nos ha solicitado una atención especial en cuanto a hacer los sitios accesibles, creemos que tanto por motivos legales como éticos, hemos de hacer que todas nuestras páginas sean lo más accesibles posible.

7.3.7 Tests sobre usuarios reales

En el caso de que sea necesario (intentaremos que no lo sea). Tendremos que hacer pruebas sobre usuarios reales. Hasta ahora sólo lo hemos hecho en determinadas ocasiones:

- En interfaces de administración.
- Con clientes que lo solicitaban.
- Con clientes que además son los propios usuarios de la aplicación.
- Mediante herramientas de escritorio remoto.

Ver el proceso de trabajo es la mejor forma de ver si lo están haciendo correctamente y si la aplicación web se adapta a su forma de trabajar.

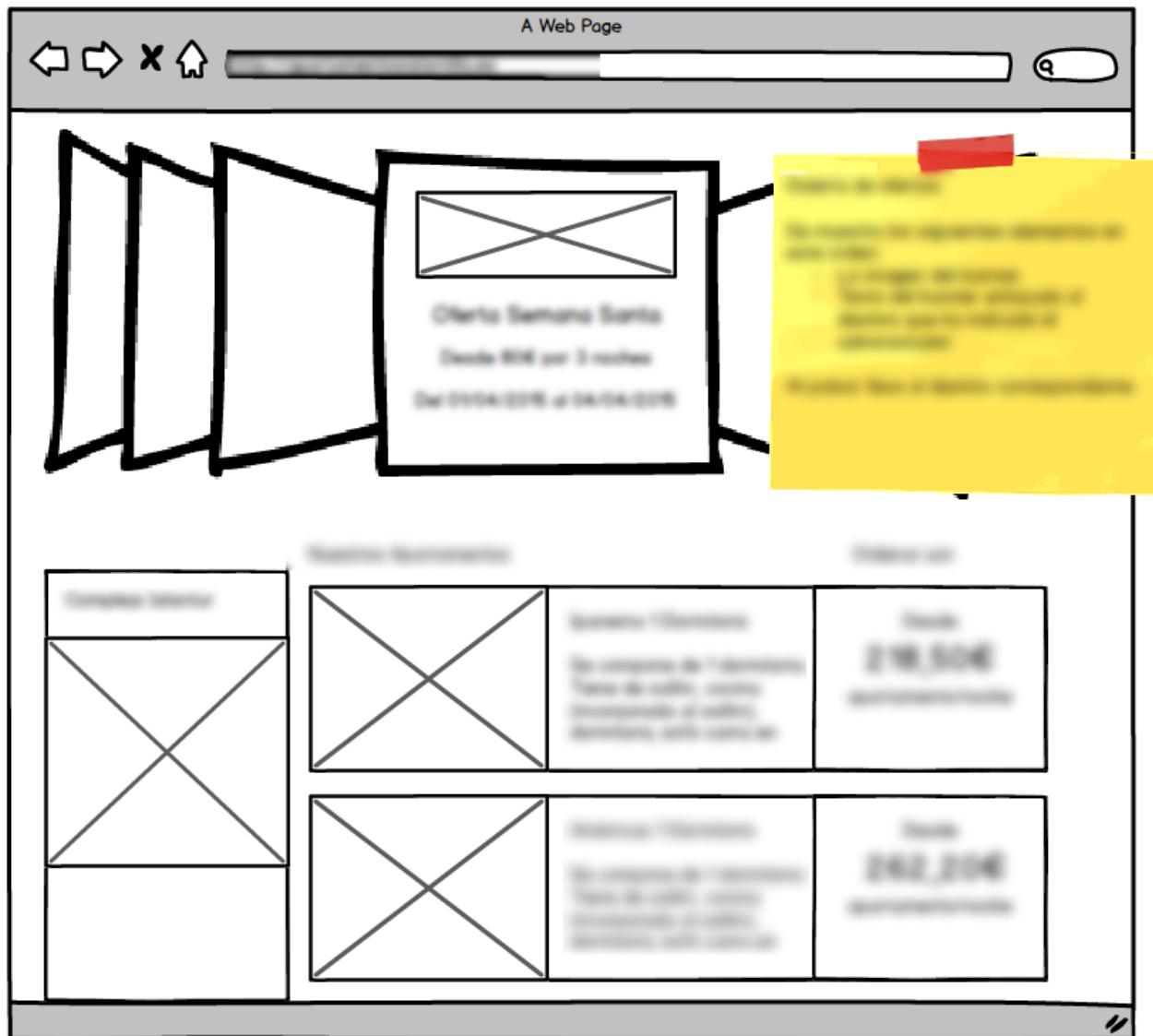
7.4 Herramientas

7.4.1 Balsamiq

En la empresa ya se usaba la herramienta web Balsamiq [BALSQ] que nos permite definir *storyboards* para los proyectos de nuevos clientes.

El problema es que no hemos usado esta herramienta para nuevos módulos una vez que el proyecto está desarrollado. Esto hacía que las interfaces

A continuación mostramos un *mockup* generado con Balsamiq:



7.4.2 Evaluadores de accesibilidad

Son páginas que proporcionan ayuda en cuanto a la evaluación de la accesibilidad. Esto es, indican qué elementos no cumplen cada una de las directrices del WCAG.

Nosotros nos basaremos principalmente en los siguientes:

- **Tota11y** que es una herramienta para la evaluación de la accesibilidad de una página web. Es muy nueva y ha sido desarrollado por la Khan Academy [TOTA11Y].
- **TAW Online**: una herramienta de evaluación de la accesibilidad web [TAWDIS] desarrollada por la Fundación CTIC [CTIC].

Por supuesto, no hay que olvidar que estas herramientas sólo pueden probar una serie de normas de forma automática. El resto habrá que seguir analizándolas de forma manual.

7.4.3 Herramientas de escritorio remoto

En el caso de que un cliente no sepa cómo realizar una acción, vamos a proponerle la instalación de una herramienta de escritorio remoto (VNC, NoMachine, TeamViewer, etc.).

Estas herramientas de escritorio remoto permiten visualizar en tiempo real cómo trabaja el cliente en directo lo que nos puede servir para simular un test de usabilidad centrado en un tipo de usuario (administrador, en la mayoría de nuestras aplicaciones).

8 Planificación

El proceso comenzó el 15 de Febrero del 2016 y espero que termine el 31 de Mayo.

La idea es implantar los cambios sobre el método del trabajo cada mes. Esto permite que el equipo se adapte y también que me informen de los progresos y su impresión sobre el método.

Los periodos no son fijos, en cuanto a que si vemos que necesitamos más tiempo para estudiar las herramientas o mejorar en un aspecto determinado, lo haremos.

Con respecto a la formación que les estoy impartiendo, entiendo que no puedo obligar a la gente a estudiar en su tiempo libre (ni espero que lo hagan). Pero me gustaría hacer de ellos unos mejores desarrolladores, no sólo en el ámbito de las metodologías ágiles, sino en el desarrollo de software en su conjunto.

8.1 Primer mes: Kanban y Pomodoro

Comencé implantando Kanban y el método Pomodoro en mi equipo con la intención de hacerles centrarse en el desarrollo y gestión de incidencias de sus proyectos. Cuatro semanas es más que suficiente para propiciar la adaptación al método y saber si es el adecuado.

Mi objetivo es hacer también tomar conciencia de que hemos de fijar un máximo número de tareas sobre las que trabajamos (límite de *WIP*, *Work In Progress*), para evitar todo lo que se pueda los cambios de contexto.

Para mejorar aún más en el desarrollo de esta transición a los métodos ágiles, vamos a cursar un curso multitudinario en línea, [GARZAS16], sobre éstos en nuestro tiempo libre.

El objetivo es implantar una cultura de que hay que ser eficientes en el entorno laboral y dar una respuesta rápida a las necesidades del cliente mediante el uso de prototipos que se van refinando poco a poco.

En definitiva, el primer mes se comenzará con la implantación del tablero Kanban y se irá preparando el camino para seguir las recomendaciones de [ANDERS10] sobre aplicar *kaizen* (o mejora continua) al equipo al completo.

8.2 Segundo mes: tests y revisiones de código

Hasta mediados de Abril mes haré énfasis en la calidad del código y el uso de tests.

En ese mes comenzaremos los tests de unidad y los tests de integración. El objetivo es que no se pase a producción ninguna funcionalidad que no se haya probado. Trataremos de realizar pruebas con *end-to-end* con *Selenium* y pruebas de unidad.

Hay un libro de pruebas con Django [PERCIVAL15] que usaremos como guía para implantar TDD en los proyectos que hagan uso de este lenguaje (todos lo que desarrollamos con el *framework* Django, obviamente).

También nos centraremos en el código limpio⁹, para ello usaremos las heurísticas de *Clean Code*, [MARTIN08] y trabajaremos algunas partes de *Refactoring*, [FOWL99]. *Clean Code* se centra especialmente en la producción de código mantenable y de calidad, mientras que *Refactoring* se centra en cómo mejorar código que ya existe, lo que en nuestro caso se aplica al cien por cien.

El objetivo de esta fase es la de generar código de alta calidad que sea fácilmente mantenible y que haga fácil la extensión de funcionalidad en los proyectos. Las pruebas nos ayudarán a permitir refactorizar más rápido y sabiendo que efectivamente funciona nuestra solución.

En [ANDERS10] David J. Anderson indica varias veces que lo primero es hacer software de calidad antes de embarcarse en la mejora de un equipo de trabajo. Por lo tanto, es importante introducir mecanismos que nos aseguren alta calidad en el código al principio del proceso.

8.3 Tercer mes: usabilidad

Desde mediados de Abril a mitad de Mayo nos centraremos en la usabilidad. Este es un aspecto que en, general en la empresa en la que trabajo se ha descuidado bastante por lo que hay mucho espacio para mejorar.

En primer lugar, estudiaremos una serie de prácticas básicas de usabilidad. El libro *Don't make me think* [KRUG14] es un clásico que todo desarrollador ha de haberse leído al menos una vez en su vida.

El objetivo es comunicarnos mediante bocetos más que mediante diagramas. Es complicado que un cliente entienda un documento de especificación, pero es mucho más sencillo que vea un *Storyboard* o varias interfaces y comprenda cuál es la funcionalidad que se va a desarrollar.

De esta forma, cada vez que vayamos a una reunión con un cliente o tengamos que explicarle una funcionalidad, abogaré por el uso de bocetos y herramientas.

8.4 Cuarto mes: mejorar la comunicación con los clientes y final

Desde finales de Mayo a Junio, si tenemos oportunidad, aplicaremos las técnicas de *Inception* y *Roadmapping* para la extracción de requisitos y su priorización con el cliente.

En este paso buscaremos realizar reuniones con el cliente en vez de enviar toda la información por correo y que el cliente la valide, tal y como hacemos ahora.

El objetivo es hacer más reuniones pero más cortas. La mayoría de los proyectos que tenemos ya están implantados y sólo requieren ampliaciones, por lo que una reunión de un par de horas es adecuada para extraer la funcionalidad de un módulo o una aplicación concreta.

En el caso de proyectos completamente nuevos sí que podremos hacer reuniones más largas con el objetivo de tener una idea más panorámica de lo que quiere el cliente. En este caso, una buena

⁹ Al menos uno de nuestros clientes es otra empresa de desarrollo de software, por lo que entregar un código limpio es lo mínimo. Para el resto de proyectos, hacer un código de alta calidad hará más fácil su mantenimiento.

forma de proceder sería la de hacer un *Inception Deck*, tal y como lo indica Jonathan Rasmusson en *The Agile Samurai*. Por ahora este caso no se ha presentado.

En el mes de Mayo procederemos a una revisión de todo el proceso y a potenciar aquellos aspectos que no hayan quedado claros.

También, si algún mes se retrasa, podremos usar estos días para, de alguna forma, tener cierto margen antes de dar por terminada esta implantación de nuestro marco de trabajo ágil.

Por último, realizaré unas encuestas a distintos implicados en el proceso para conocer su opinión sobre nuestra nueva forma de trabajar y el estado del proceso de implantación de Kanban.

8.5 Futuro

Evidentemente mi objetivo es que este proceso no termine aquí y podamos seguir trabajando en este marco de desarrollo ágil en la empresa. Parece que, hasta cierto punto, gozo del apoyo de la dirección, por lo que podré seguir aplicando técnicas ágiles y por tanto, mejorando nuestro desempeño en el desarrollo de software y la satisfacción de los clientes.

Me gustaría continuar el proceso de mejora de manera que pudiéramos evaluarlo cada seis meses. Para ello podríamos hacer uso de cuestionarios u otras medidas más objetivas sobre los tiempos de desarrollo de las tareas.

Otro aspecto sobre el que me gustaría seguir trabajando es el *kaizen*. Hay que seguir motivando a los desarrolladores para que crezcan y puedan aportar cada vez más al equipo. Por el bien de la organización a la que pertenecen y por el suyo propio a lo largo de su carrera profesional.

9 Evaluación

9.1 Introducción

Hay tres grandes tipos de evaluaciones que se pueden hacer cuando se está en un proceso de mejora:

- Evaluar si el sistema está bien implantado
- Evaluar la mejora alcanzada, detectando restricciones en el flujo de trabajo ir solucionándolos. Para ello usaremos varias métricas ampliamente conocidas sobre Kanban.
- Evaluar si el sistema está teniendo buena acogida entre los interesados (desarrolladores, dirección y dueños del producto).

Nosotros vamos a realizar estas tres evaluaciones. Primero veremos si nuestra implantación ha sido un éxito (y en qué grado), después evaluaremos con valores objetivos el rendimiento, y una vez que esté implantado evaluaremos de forma objetiva el curso de cada proyecto.

En un futuro, mejoraré [PYSTTRLO] para obtener informes automatizados, de manera que el equipo sepa el estado de su proyecto en un instante dado. Después, volveré a evaluar la implantación para ver si (al menos el apartado de visualización) ha mejorado.

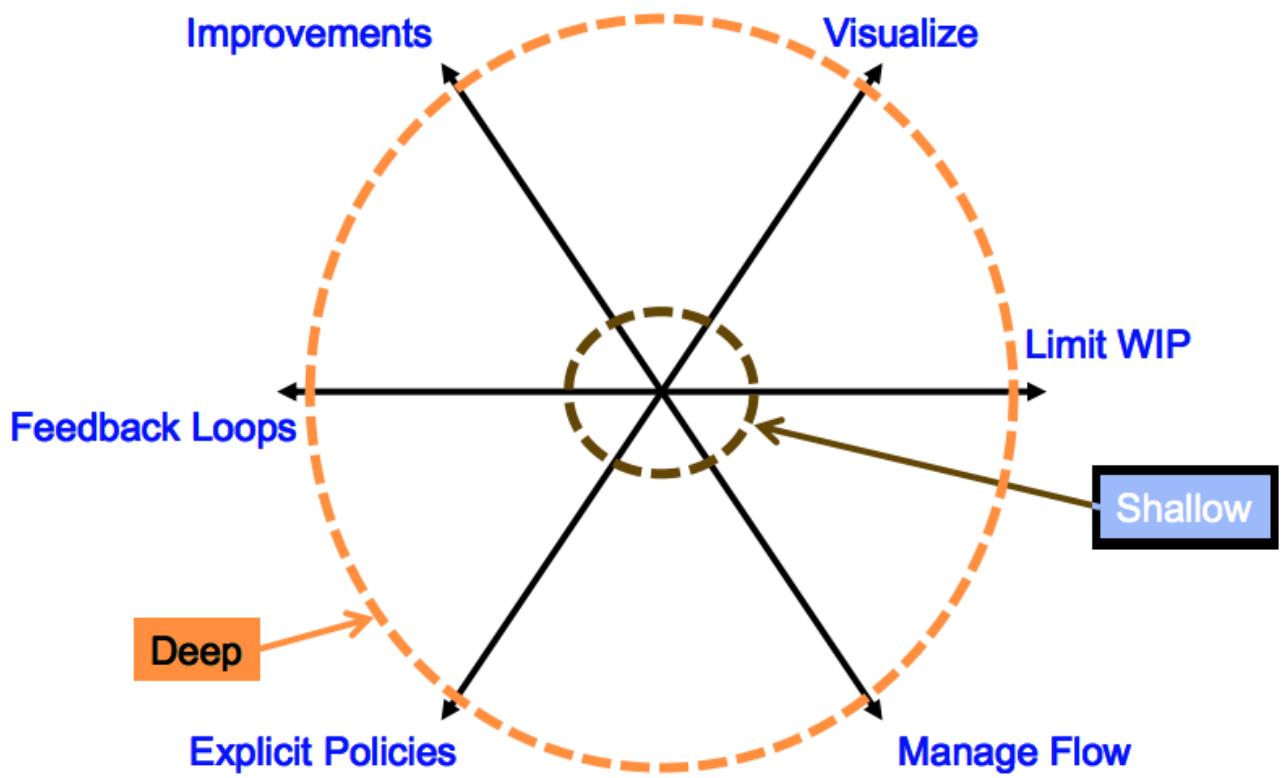
Sin más dilación, comenzamos describiendo la que será la primera evaluación: la evaluación de la implantación de Kanban.

9.2 Evaluación de la implantación

Para evaluar la implantación, vamos basarnos en las 6 prácticas principales para lograr una adopción Kanban con éxito de David J. Anderson descritas en *How deep is your Kanban?* [ANDERS12]:

1. Visualización
2. Limitar el número máximo de tareas (WIP)
3. Gestionar el flujo de trabajo
4. Hacer las políticas de trabajo explícitas
5. Implementar retroalimentación
6. Mejorar la colaboración y evolucionar de forma experimental

En [ANDERS12] se describe un diagrama de estilo *Kiveat* para visualizar la calidad de la implementación en cada una de estas 6 dimensiones.



(c) David J. Anders & Associates, Inc.

Para evaluar cada una de estas dimensiones, vamos a realizar un cuestionario en el que con preguntas sencillas se realicen preguntas a los desarrolladores para que muestren su visión del estado de cada una de estas dimensiones.

Mostramos a continuación el cuestionario que se ha pasado a los desarrolladores de cada proyecto:

Mejora (10)	<p>1. Sabemos cómo hacer las cosas bien.</p> <p>2. Hay retrovisivas frecuentes.</p> <p>3. El equipo sabe cuál es su rendimiento.</p> <p>4. El equipo sabe cuáles son sus debilidades.</p> <p>5. El equipo sabe cuál es el estado del proyecto.</p> <p>6. El equipo conoce lo que falta para terminarlo.</p> <p>7. El equipo conoce los obstáculos del proyecto.</p> <p>8. El equipo sabe en qué obstáculos se están trabajando.</p> <p>9. El equipo sabe los pasos para resolver obstáculos.</p> <p>10. El equipo sabe lo que ha aprendido en cada momento.</p>	Efectos (evidencias de que esto funciona [12])	<p>1. Los miembros del equipo ven y entienden la panorámica de cada proyecto.</p> <p>2. Hay sentimiento de equipo (ayuda y respeto).</p> <p>3. El foco está en eliminar las tareas que bloquean.</p> <p>4. Se terminan las tareas actuales antes de comenzar otras.</p> <p>5. El equipo tiene prioridades y las cumple.</p> <p>6. No hay estrés (el equipo trabaja a un buen ritmo pero no excesivo).</p> <p>7. Hay motivación para buscar mejoras en el software y el trabajo.</p> <p>8. Hay evolución del proceso (cambios en WIP, flujo de trabajo, visualización, etc.) Para adaptarse mejor al proceso.</p> <p>9. Ha habido un incremento de la profundidad de la implantación Kanban.</p> <p>10. El proceso de evolución sigue el modelo ágil Kanban.</p> <p>11. Hay mentorización y tiene efecto en el proceso de mejora.</p> <p>12. El comportamiento de los flujos de trabajo hace que el proceso evolucione. Nos adaptamos al trabajo (no al revés).</p>	Visualización (13)	<p>1. Se ven las tareas sobre las que se está trabajando.</p> <p>2. Se ven los tipos de tareas de forma diferenciada.</p> <p>3. El flujo de trabajo es visible.</p> <p>4. Se ve las tareas siguientes y cuáles están terminadas.</p> <p>5. Se ve sobre lo que está trabajando cada miembro.</p> <p>6. Se indican los bloqueos de alguna forma.</p> <p>7. Las políticas de trabajo están visibles en algún momento.</p> <p>8. Hay un estado "Listo y pendiente de desplegar".</p> <p>9. Las métricas (cycle, lead, S/E, etc.) son visibles.</p> <p>10. Los límites WIP están visibles para cada estado.</p> <p>11. Las dependencias entre tareas (jerarquía de tareas) se muestran.</p> <p>12. Se muestran las dependencias entre flujos de trabajo.</p> <p>13. Se muestra el riesgo de cada tarea y característica (fecha límite de tarea, coste de retraso, compromiso con el cliente, riesgos técnicos o de mercado).</p>
Retroalimentación (7)	<p>1. Hay reuniones diarias.</p> <p>2. A los implicados de cada proyecto (cliente, equipo comercial, etc.) se les informa con frecuencia.</p> <p>3. Dirección conoce el estado del equipo y de los proyectos.</p> <p>4. Conversaciones con otros miembros del equipo sobre problemas del proyecto.</p> <p>5. Conocimiento de la situación de facturación del proyecto.</p> <p>6. Presentaciones y discusiones sobre KPI (ratio de defectos, satisfacción del cliente).</p> <p>7. La frecuencia de este tipo de presentaciones y discusiones es al menos una vez al mes.</p>	Políticas de trabajo claras y en uso (12)	<p>1. ¿Hay distintos tipos de tareas?</p> <p>2. ¿Se indica la prioridad de las tareas?</p> <p>3. ¿Hay alguien que gestiona las colas de tareas cada cierto tiempo?</p> <p>4. ¿Hay un proceso de asignación de trabajo?</p> <p>5. Hay una definición de preparada para entrar en desarrollo.</p> <p>6. Hay un proceso de estimación definido.</p> <p>7. Se limita el tamaño de las tareas.</p> <p>8. Está claro cómo seleccionar tareas de un estado para moverlas a otra.</p> <p>9. Hay una estrategia para compartir conocimiento.</p> <p>10. Hay definición de "Terminado" en cada estado.</p> <p>11. Hay clases de servicio (¿se gestionan de forma distinta las tareas según su naturaleza y riesgo?)</p> <p>12. Se sabe cuál es la capacidad máxima de trabajo por cada clase de servicio.</p>	Límite de WIP (4)	<p>1. No hay límite WIP pero sí un compromiso en terminar las tareas empezadas antes de empezar con las nuevas. Trabajando así con un máximo de tareas cómodo.</p> <p>2. Hay límites WIP explícitos; por persona y en algunas columnas.</p> <p>3. Hay límite WIP a nivel de flujo de trabajo. Esto es, a nivel de características del sistema desarrollando.</p> <p>4. Hay muchos flujos de trabajo independientes en función de quién solicita la tarea y si viene del cliente o es una necesidad interna.</p>
Manage Flow (11)	<p>1. Reuniones diarias de planificación.</p> <p>2. Los problemas extremos se paran y se escalan (al líder de equipo, luego a la dirección, etc.)</p> <p>3. La siguiente tarea se reprioriza continuamente en función de las necesidades del cliente.</p> <p>4. Se actualiza el documento con la cuenta de horas al terminar cada tarea.</p> <p>5. El equipo sabe el reto/problema del proyecto en el que está trabajando.</p> <p>6. ¿Se sabe el objetivo del proyecto y cuándo ha sido completado?</p> <p>7. El equipo sabe el obstáculo sobre el que se está trabajando.</p> <p>8. El equipo conoce los obstáculos que impiden llegar al objetivo.</p> <p>9. El equipo sabe los pasos para resolver cada obstáculo.</p> <p>10. El equipo aprende en cada obstáculo.</p>	Visualized (13)		Limit WIP (4)	<p>1. Reuniones diarias de planificación.</p> <p>2. Los problemas extremos se paran y se escalan (al líder de equipo, luego a la dirección, etc.)</p> <p>3. La siguiente tarea se reprioriza continuamente en función de las necesidades del cliente.</p> <p>4. Se actualiza el documento con la cuenta de horas al terminar cada tarea.</p> <p>5. El equipo sabe el reto/problema del proyecto en el que está trabajando.</p> <p>6. ¿Se sabe el objetivo del proyecto y cuándo ha sido completado?</p> <p>7. El equipo sabe el obstáculo sobre el que se está trabajando.</p> <p>8. El equipo conoce los obstáculos que impiden llegar al objetivo.</p> <p>9. El equipo sabe los pasos para resolver cada obstáculo.</p> <p>10. El equipo aprende en cada obstáculo.</p>

Este cuestionario está basado en el trabajo de Christophe Achouiantz [ACHO13]. Puede verse una copia presente en el anexo 1.

No he puesto ningún límite por niveles (colores) de cada una de las dimensiones de mejora de Kanban descritas por Achouiantz, ya que cada color representa un nivel, un reto que han de superar los equipos antes de embarcarse en el siguiente.

Adaptado de
<http://leanagileprojects.blogspot.com.es/2013/03/depth-of-kanban-good-coaching-tool.html>
 (c) Christophe Achouiantz

Dado que esta primera evaluación también de alguna forma debía seguir como motivación al equipo, tampoco quería someter al equipo a un estrés innecesario, aunque sí que quería que me dieran su opinión sobre cada uno de los proyectos.

9.3 Evaluación de la mejora alcanzada

Dado que no hay datos objetivos anteriores sobre los resultados obtenidos con el método de trabajo “tradicional”, va a ser complicado evaluar la mejora de forma clara y objetiva.

Este marco ágil y sus buenas prácticas que quiero implantar es muy polivalente, pero necesitamos saber si se mejora la productividad del equipo de desarrollo.

Por eso, a continuación detallo varias métricas que pueden ayudarnos a saber si el equipo está teniendo un rendimiento mayor o no.

Estas métricas se calculan con PyStats-Trello [PYSTTRLO] y hay una fichero con los resultados de un ejemplo en el anexo 2.

9.3.1 Métricas

Kanban tiene varias métricas que son adecuadas para medir el rendimiento del equipo.

Trello, la herramienta web que estamos usando no proporciona estas herramientas de serie, por lo que tendremos que buscar complementos o desarrollar nuestras propias utilidades que nos permitan obtener las siguientes métricas:

9.3.1.1 Lead

Media de los tiempos que está en espera la tarea hasta que se entrega al cliente el nuevo desarrollo (o la resolución de incidencia). Esto es, tiempo que hay entre que entra al tablero y pasa al estado “Terminado”.

Una forma de ver esta métrica es pensar que es el tiempo de espera del cliente desde que ha solicitado el cambio. A fin de cuentas, el que la tarea esté en un estado u otro es indiferente para el cliente, ya que sólo ve el resultado final, esto es, cuando la tarea se termina completamente.

Nosotros mediremos la media (y su desviación) de estos tiempos de espera hasta el comienzo del desarrollo de todas las tareas para cada tablero.

9.3.1.2 Cycle

Tiempo de desarrollo de la tarea. O dicho de otro modo, tiempo desde que entra en la columna “En desarrollo” hasta que pasa a la de “Terminado”.

Nosotros mediremos la media (y su desviación) de los tiempos de desarrollo de las tareas para cada tablero.

9.3.1.3 Tiempo medio por estado

Calcular la media y la desviación de lo que tarda las tareas en cada columna es una forma de

detectar cuellos de botella. Así, de esta forma podemos averiguar qué proceso es el que está retrasando el desarrollo del proyecto. Si el desarrollo no es el estado que tiene un mayor tiempo las tareas, entonces es que tenemos un proceso poco eficiente a nivel organizativo.

Por otro lado, dado que hay varios equipos implicados (según el estado en el que se encuentre), podemos ver el desempeño de cada uno de los equipos en este proyecto. De esta forma, podremos detectar si necesitan de más recursos o si necesitan formación adicional para realizar con más destreza su parte del flujo de trabajo.

9.3.1.4 Número de vueltas atrás por estado

En todos los estados, volver atrás significa que no hemos realizado el proceso todo lo bien que deberíamos hacerlo hecho. En especial, si se vuelve a “En desarrollo”, implica que la tarea no es correcta, bien por fallos, o bien porque no cumple con los requisitos.

Contar el número de vueltas atrás que se produce para cada uno de los estados de las tareas de un proyecto es importante porque indica que hay tareas que no las hacemos lo suficientemente bien y que en el momento de pasar al siguiente proceso, no tienen la calidad necesaria. Bien porque hablamos de software y falle, o bien porque no se ha tenido una conversación con el cliente para validar que la tarea era correcta o, en definitiva, por cualquier motivo.

Por ejemplo, si vemos que una tarea en concreto no para de volver hacia atrás, seguramente tenga una complejidad mayor de la esperada y el equipo no sea capaz de tratar con ella, por lo que se aconseja dividirla y tratarla como tareas más sencillas hasta encontrar dónde reside la complejidad.

9.3.1.5 Tiempo estimado de desarrollo

El tiempo estimado de desarrollo es el tiempo, en horas, que un desarrollador estima que tardará en completar la tarea.

El tiempo ha de ser razonable por lo que se potencia que los desarrolladores dividan las tareas grandes en tareas de un tamaño más pequeño (no más de 1 o 2 días), de manera que puedan estimarlas con el menor desfase posible y por ende, nuestros presupuestos serán los más reales.

No usamos ningún método de estimación más allá de la experiencia de cada uno de los miembros del equipo. Para estimaciones de épicas sí que hemos hecho uso de métodos como puntos objeto o incluso de parte del método de puntos de función. A veces, sí hacemos uso de estimaciones a pares, de manera que la estimación sea consensuada por los miembros del equipo, defendiendo cada uno su postura basada en su experiencia.

Si bien algunos equipos hacen uso de puntos de historia como unidades de medida, este concepto se muestra poco práctica en nuestro caso. Al cliente final se le cobrará no por las horas desarrolladas, sino por éstas ponderadas según la naturaleza del proyecto y desarrolladores, así como su número, claro está.

El tiempo estimado de desarrollo lo mediremos por tarea, usuario y período.

9.3.1.6 Tiempo efectivo de desarrollo

Es el tiempo efectivo consumido en la tarea (en horas). Como estoy promoviendo técnicas de concentración y de compartmentalización de tiempo, este tiempo ha de ser lo más cercano al real posible.

El tiempo efectivo de desarrollo lo mediremos por tarea, usuario y período.

También mediremos la diferencia entre los tiempos estimados y desarrollados, para, de esta forma, conocer más información sobre qué desarrolladores tienen conocimientos suficientes y realizan buenas estimaciones, y en qué períodos de tiempo hicimos malas estimaciones, ocasionando por ello, un coste para la empresa.

9.3.1.7 Velocidad de desarrollo semanal

Una forma de evaluar el desarrollo es observar la progresión de las horas que destina cada uno de los miembros de mi equipo a desarrollar a lo largo de una semana.

El objetivo es que estas horas sean un 70% del tiempo al tener menos interrupciones y una disciplina de trabajo más adecuada.

Usar la velocidad de desarrollo como medida absoluta y única de rendimiento del equipo puede ser perjudicial porque no indica de ninguna forma que lo que se está desarrollando es lo correcto.

También, envía un mensaje equivocado a los miembros del equipo: “cuantas más horas de desarrollo mejor”, por lo que pueden pensar que se incentiva el que los empleados hagan horas extras sin cobrar¹⁰.

Por otro lado, fomentar una cultura de no buscar el valor sino simplemente muchas horas de trabajo es también erróneo. Un desarrollador que trabaje menos horas puede ser mucho más productivo y generar código de mejor calidad que otro.

Por último, puede que haya situaciones en las que haya tareas necesarias para el proyecto que consuman muchas horas que no se pueden repercutir en el desarrollo. De esta forma, la velocidad de desarrollo baja, como ocurre por ejemplo en el caso de la formación. A veces se le puede imputar a un proyecto, pero si es una funcionalidad o tecnología general, no se podrá imputar a ningún proyecto concreto.

9.3.1.8 Número de tareas por hora

Dado que uno de los proyectos de mi equipo ha estado 6 meses sin realizar ninguna, una buena forma de saber si estamos creando una cultura ágil es la de contar el número de entregas que se hacen al cliente al mes.

Debemos tener en cuenta que en algunos proyectos ya implantados es más sencillo realizar entregas, si entendemos como entregas cada una de las resoluciones de incidencias al cliente. Pero

¹⁰ Me consta que esta práctica es común en grandes compañías. Se indican unos objetivos totalmente irreales, forzando a trabajar más de las 8 horas diarias. Teóricamente, los empleados que los consiguen ascienden.

en los proyectos nuevos de desarrollo sí que nos podrá servir como medición de que el proyecto va recibiendo retroalimentación por parte del cliente.

También es cierto que será un reto medir el número de número de entregas en algunos proyectos debido al gran número de entregas.

Además, ¿es el número de tareas una buena medida? Sólo si todas las tareas tienen el mismo valor. Es cierto que tener una tasa de entregas alta puede ser beneficioso de cara a dar una confianza en mantener una respuesta constante a las necesidades del cliente. Pero debemos ser vigilantes con el riesgo de caer en un descenso de la calidad del desarrollo de software para entregar lo máximo posible.

Como Kanban no tiene un concepto de valor por tarea más allá de la urgencia y las herramientas que estamos usando tampoco nos dan un apoyo para extraer información, no vamos a tener esta medida en nuestro trabajo.

9.3.1.9 Número de incidencias por proyecto

Actualmente no registramos las incidencias de los clientes, pero una forma de saber si los clientes están notando una mejora en la calidad es la de registrar el número de incidencias, por dos motivos:

En primer lugar, las incidencias pueden nacer de fallos en el software. Con este proceso de mejora buscamos implantar una mejor calidad en el código. Por lo que debería haber una bajada del número de este tipo de incidencias.

Por otro lado, hay incidencias que están basadas en fallos de usabilidad. Interfaces complejas que no representan adecuadamente los procesos del negocio son la mayor causa de incidencias (según mi propia experiencia). El uso de bocetos y el seguimiento de las prácticas heurísticas de usabilidad debería reducir este tipo de incidencias.

Una forma sencilla de calcular el número de incidencias es asignar una etiqueta en Trello a las tareas que sean incidencias. PyStats-Trello nos indicará en qué período de tiempo se han creado. Si conforme avanza el tiempo del proyecto el número de incidencias baja, es que se está desarrollando software de mejor calidad.

9.3.2 Herramientas para realizar las mediciones

Lamentablemente Trello (al menos en su versión gratuita) no tiene un análisis estadístico con gráficas y resúmenes disponibles para los usuarios de su aplicación web.

Por eso, he desarrollado un pequeño software en Python que realiza las mediciones anteriormente detalladas.

El proyecto se llama PyStats-Trello y es código abierto y está disponible a libre disposición de todos los desarrolladores o ingenieros que lo deseen en [PYSTTRLO].

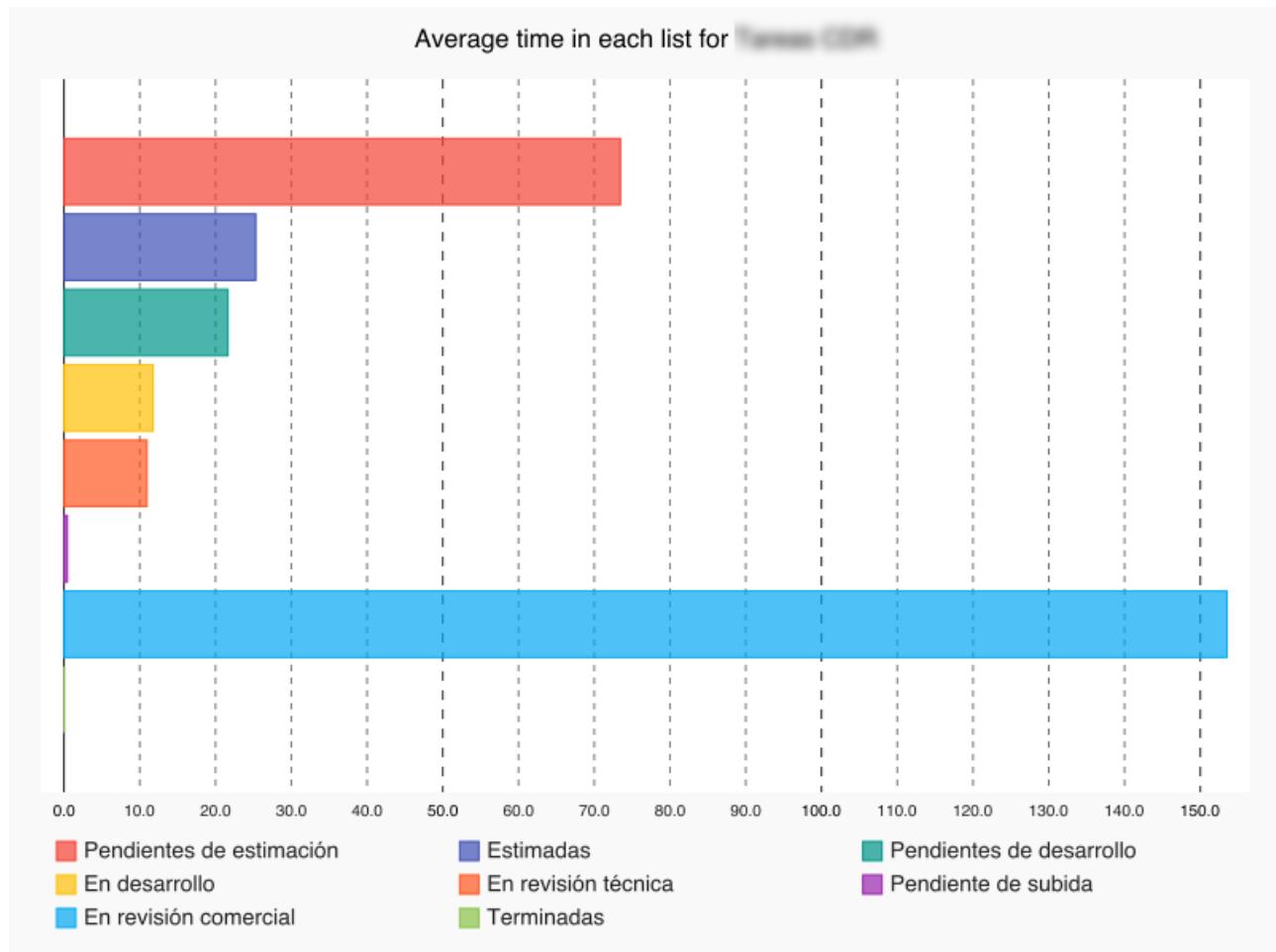
Este software hace uso de una librería de consulta a la API de Trello desarrollada por voluntarios. Para poder ser más eficiente, he realizado varios cambios sobre su proyecto y mi copia del proyecto

puede verse aquí en [PYTRELLO].

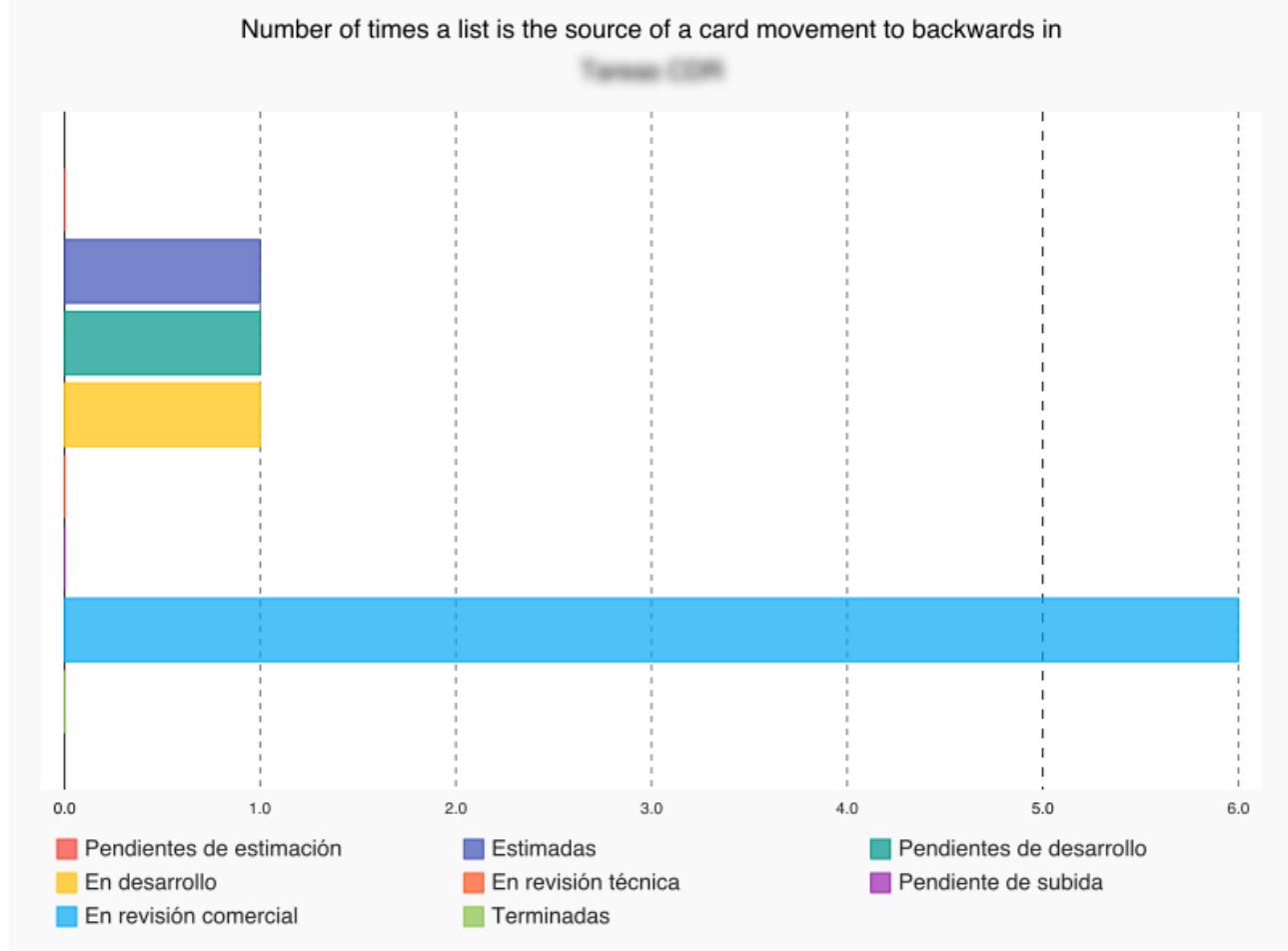
En el momento de la confección de esta memoria se ha solicitado la mezcla de mis cambios al proyecto Py-Trello original pero todavía no se han integrado con la rama principal.

El software genera tanto una salida con gráficas de resumen de los tiempos medios de las tareas por columna como la cuenta de las veces que los movimientos de las tareas son hacia adelante o hacia atrás.

Por ejemplo, la siguiente gráfica muestra para un proyecto la media del tiempo (en horas) de las tarjetas en cada columna. Como podemos ver, hay un problema de gestión con el equipo comercial porque no termina de revisar en un plazo razonable las tareas que desarrolla el equipo de desarrollo.

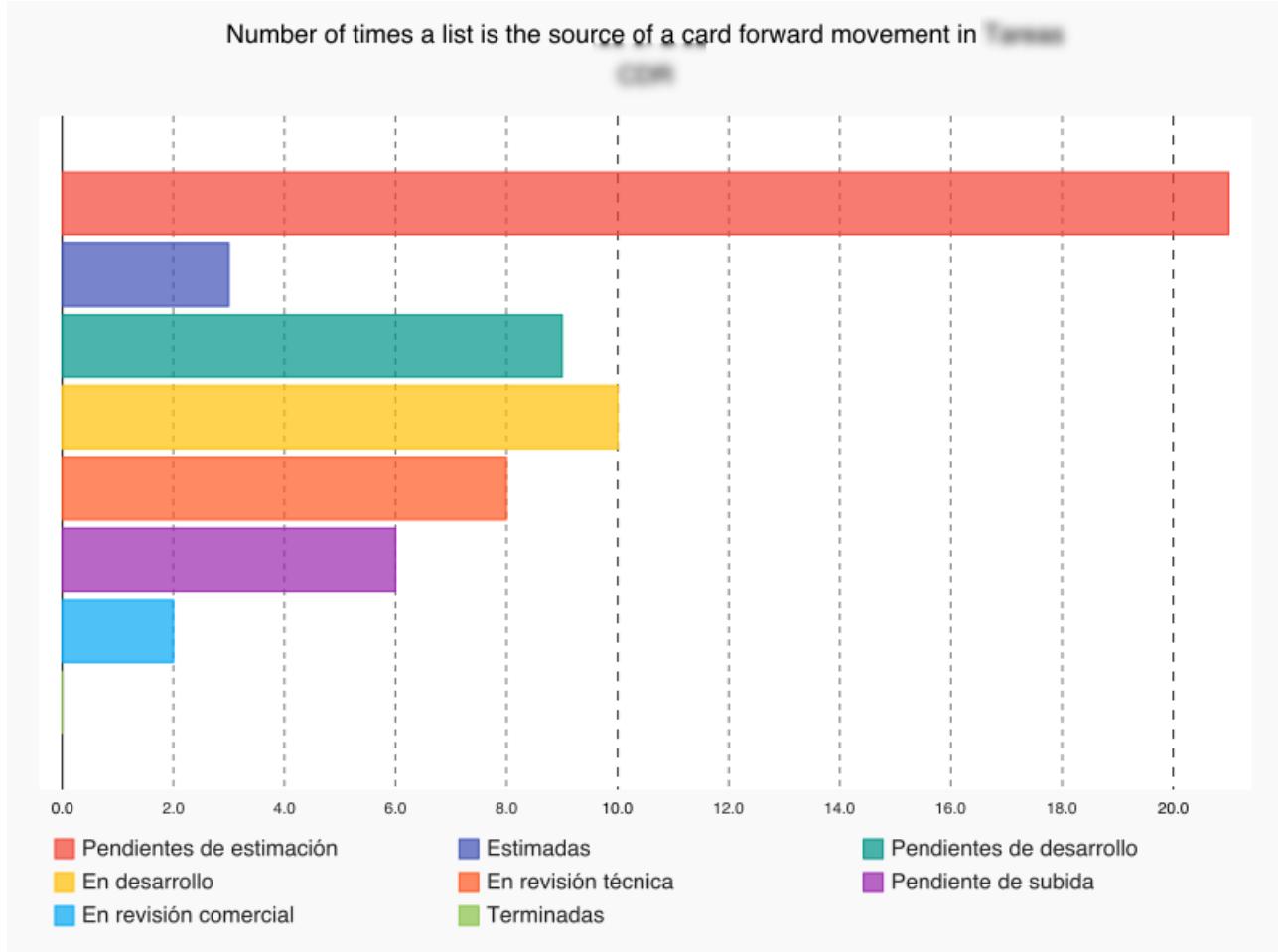


En el caso de querer ver el número de tareas que son enviadas al estado anterior, podemos ver que el software que he desarrollado genera un gráfico de la siguiente forma:



Aquí vemos como el equipo comercial es también muy exigente y es el que más ha devuelto tareas a otros estados en el proceso de de revisión. El resto de estados tienen una tasa de devolución a otro estado prácticamente despreciable (una o ninguna tarea).

El software también genera un gráfico con información de cuáles son los estados de los que más se mueve hacia delante, o lo que es lo mismo, los estados de los que es más fácil partir:



Es previsible que el estado del que más se avance sea el primero, que es justo lo que se sucede en este ejemplo. En este caso concreto parece que muchas tareas no pasan por la estimación. Esto es, quizás, porque en este proyecto concreto tenemos mucha experiencia y no necesitamos prácticamente estimar las nuevas funcionalidades. Sea como sea, analizaremos estas cuestiones en los dos procesos de análisis que hemos hecho sobre el desarrollo y que se encuentran en capítulos posteriores.

9.3.3 Mediciones subjetivas

9.3.3.1 Número aproximado de interrupciones

Según el estudio de Parmin y Rugaber [PARMIN2012], volver a concentrarse después de una interrupción consume unos 15 minutos. Si contamos con que cada desarrollador suele ser interrumpido unas 4 veces al día, ya tenemos una hora de trabajo perdida al día. Esto hace unas 20 horas de trabajo al mes, lo que es igual a casi 3 días al mes perdidos.

Con este marco ágil que estoy implantando, busco reducir interrupciones y así, el número de tiempos perdidos. Mi objetivo es mejorar el rendimiento de los desarrolladores haciendo que se centren cada día en una única cosa: desarrollar.

Así, medir el número de interrupciones (aproximado) es una buena forma de saber si se están implantando este marco ágil de trabajo correctamente.

9.3.3.2 Cuestionarios

Si bien no son objetivos, suponen una gran herramienta para conocer de forma rápida la opinión de cada uno de los implicados en el proceso de desarrollo sobre la implantación de la metodología ágil.

La diferencia entre estos cuestionarios con los de la profundidad de Kanban es que estos cuestionarios no tratan de saber cómo de bien se están implantando Kanban, sino si el equipo, los dueños del producto y la dirección están contentos con el resultado.

Puede ser que la implantación de un marco de trabajo sea excelente, pero no esté teniendo éxito y el resto de implicados en los proyectos no tengan una buena opinión sobre esto. Por ejemplo, si no están acostumbrados a dar retroalimentación, o si el proyecto tiene unos requisitos estáticos fijados por un pliego de condiciones (y por tanto no se adapta a un entorno ágil), etc.

Estos cuestionarios son originales pero están basados en los de [RIBEIRO05], [WATERS08] y en especial los que se usaron en Valpak [VALPAK13] para evaluar los dos años de evolución con un marco de desarrollo ágil. Para más información, en [LINDERS16] hay un compendio de cuestionarios sobre marcos de trabajo ágil (en especial Scrum).

Los cuestionarios están diseñados de manera que todos los enunciados sean afirmativos y se evalúen de 1 (muy en desacuerdo) a 5 (muy de acuerdo). Así, sólo habrá que sumar la puntuación de todas las preguntas y cuanta más puntuación obtenga el cuestionario, mejor valoración habrá hecho ese individuo sobre el funcionamiento y utilidad de Kanban en el desarrollo de software.

9.3.3.2.1 Dirección

Se realizará un cuestionario a la dirección para que evalúe el proceso de conversión del equipo desde su punto de vista.

En su cuestionario se le preguntará por su valoración en cuanto a:

- Rendimiento del equipo.
- Satisfacción de los clientes.
- Mejora en presupuestos.

Mostramos el cuestionario completo a continuación:

Encuesta para la dirección sobre la implantación del proceso de desarrollo ágil en un equipo de desarrollo

Contesta cada una de las preguntas con los siguientes valores:

1 Muy en desacuerdo - 2 En desacuerdo - 3 Ni en desacuerdo ni de acuerdo - 4 De acuerdo - 5 Muy de acuerdo

1. Estoy contento con la cultura del equipo
2. Veo una mejora en la productividad del equipo
3. Veo mayor motivación en los miembros del equipo
4. El equipo me proporciona métricas adecuadas
5. El equipo me da los informes en el formato adecuado
6. La calidad en los proyectos del equipo ha mejorado
7. El equipo aporta valor a la empresa
8. El equipo aporta valor a los clientes
9. El software generado es de suficiente calidad
10. El equipo genera trabajo de forma predecible
11. Los clientes están satisfechos con este marco de trabajo.

12. ¿Qué te gustaría mejorar del manager?

13. ¿Qué te gustaría mejorar del equipo?

14. Comentarios adicionales

9.3.3.2.2 Equipo comercial y clientes

El equipo comercial es el encargado de gestionar las incidencias en varios de nuestros proyectos, por lo que es un actor muy importante en este nuevo escenario.

Se le preguntará por su percepción sobre el número de incidencias que había antes de comenzar este proceso de mejora y el número de incidencias después de haberlo concluido.

También le pediremos su opinión sobre la comunicación con el facilitador, de manera que puedan dar su opinión sobre si ha mejorado o si le ha puesto demasiadas limitaciones.

Además de ser un actor importante, el equipo comercial también actúa como dueño del producto para varios de nuestros proyectos, por lo que la encuesta para el equipo comercial y los clientes será la misma.

En el caso de los clientes, deberán evaluar nuestro trabajo y la calidad percibida de sus proyectos.

En primer lugar, deberán indicar si el tiempo de respuesta ante las incidencias es adecuado. Dado que tenemos muchos proyectos, hemos de dar respuesta a todos ellos de forma que no haya “inanición” en alguno de los proyectos.

Otro aspecto interesante es saber si el software se adapta lo suficientemente a sus necesidades. Si obtenemos una buena puntuación en este aspecto, quiere decir que estamos haciendo bien el proceso de desarrollo iterativo e incremental.

A continuación mostramos la encuesta:

Encuesta para el dueño del producto sobre la implantación del proceso de desarrollo ágil de software

Contesta cada una de las preguntas con los siguientes valores:

1 Muy en desacuerdo - 2 En desacuerdo - 3 Ni en desacuerdo ni de acuerdo - 4 De acuerdo - 5 Muy de acuerdo

1. El equipo desarrolla la funcionalidad en el orden adecuado
2. El equipo aporta valor más allá de desarrollar sólo lo que le indico
3. Veo al equipo comprometido con el proyecto
4. La herramienta que usamos es adecuada para comunicarme con el equipo
5. La herramienta que usamos es adecuada para visualizar el estado del proyecto
6. Las entregas no se retrasan
7. El tiempo de respuesta ante las incidencias es el adecuado
8. El software desarrollado tiene calidad
9. El equipo recibe retroalimentación de forma continua bien con nuevas tareas o modificaciones de tareas
10. El equipo hace caso a la retroalimentación que le doy
11. El software se adapta bien a mis necesidades
12. Hay buena comunicación con el equipo
13. ¿Qué te gustaría mejorar del equipo?

14. Comentarios adicionales

9.3.3.2.3 Desarrolladores del equipo

Hace dos años pasé unos cuestionarios para la evaluación del nivel CMMI-DEV a los jefes de equipo de la empresa en la que trabajo. Sólo uno de los ingenieros (que además era socio de la empresa) se lo tomó en serio. El resto o no contestó o lo contestó de mala gana.

Ni qué decir tiene, que las buenas prácticas (organizadas en áreas de proceso) que promulga CMMI-DEV no aparecieron en ninguno de los cuestionarios porque no estaban (ni hasta ahora están) presentes hoy en día en la empresa.

Me gustaría comprobar si la situación ha cambiado algo (al menos dentro de mi equipo). Por esto, uno de los cuestionarios que le pasaré a los desarrolladores de mi equipo será éste. Evidentemente

no aspiro a aplicar un nivel madurez muy elevado, pero como control de qué nuevas prácticas se están aplicando, creo que es buena idea.

Por otro lado, los desarrolladores deben registrar un incremento de la motivación en su día a día. Un desarrollador motivado no genera código de calidad ni sigue las recomendaciones de agilidad en sus proyectos.

Al finalizar este proceso les pasare un cuestionario (más general) en el que se les pregunte sobre la experiencia en los siguientes campos:

- Satisfacción personal
- Nivel de comunicación entre el equipo
- Nivel de aprendizaje
- Reducción del estrés

También en este cuestionario les pediré que evalúen:

- El proceso de formación recibido.
- Mi liderazgo.
- El proceso en su totalidad.
- Opinión sobre el desarrollo ágil

Por supuesto, para hacer las cosas más representativas el cuestionario será anónimo.

El cuestionario es el siguiente:

Encuesta para el equipo sobre la implantación del proceso de desarrollo ágil de software

Contesta cada una de las preguntas con los siguientes valores:

1 Muy en desacuerdo - 2 En desacuerdo - 3 Ni en desacuerdo ni de acuerdo - 4 De acuerdo - 5 Muy de acuerdo

1. La formación que he recibido sobre Kanban es la adecuada

2. Las herramientas son las apropiadas para implantar Kanban

3. La transición de la forma de trabajar se ha hecho de forma adecuada

4. Creo que podemos mejorar nuestro rendimiento aún más

5. He visto mejora en mi trabajo desde que comenzó el proceso

6. La arquitectura software es más adecuada a cada proyecto

7. Nuestro software es de mejor calidad

8. Hacemos código limpio

9. Nuestro software es más usable

10. Todo el software que hacemos está documentado

11. Hago uso de bocetos siempre que tengo que explicar una interfaz compleja

12. El equipo colabora de forma eficiente

13. El equipo ha mejorado su productividad desde que se implantó este marco de trabajo

14. Estoy viendo que mi trabajo tiene relevancia en la empresa

15. No hay picos de trabajo y el ritmo es el adecuado

16. Los cuellos de botella no son responsabilidad de nuestro equipo

17. Nos organizamos a nosotros mismos como estimamos oportuno

18. Creo que se puede visualizar en cada momento el estado del proyecto fácilmente

19. Mis compañeros de equipo me motivan

20. Mi motivación ha crecido desde que ha comenzado este proceso

21. Confío en mis compañeros de equipo para terminar proyectos

22. Siento que se me escucha en el equipo

23. Estoy contento de formar parte de este equipo

24. Todos los miembros del equipo contribuyen en el trabajo

25. No tengo interrupciones que me hacen perder tiempo en el día a día

26. El entorno es el adecuado para desarrollar (ruido, luz, comodidad...)

27. La falta de documentación nos está suponiendo un problema en algunos proyectos

28. Los clientes están colaborando activamente en el proceso de mejora

29. Otros equipos están colaborando activamente en el proceso de mejora

30. El método de cajas de tiempo mejora la productividad

31. Tenemos los recursos suficientes para seguir con el proceso de implantación

32. Compañeros de otros equipos me motivan con su ejemplo

33. Las reuniones diarias son adecuadas

34. He aprendido de las reuniones diarias

35. Las reuniones de retrospectiva son adecuadas

36. He aprendido de las reuniones de retrospectiva

37. Mi nivel de estrés se ha reducido gracias a este nuevo marco de trabajo

38. Siento que el facilitador me escucha

39. No volvería a la forma de trabajar anterior

40. ¿Qué te gustaría mejorar del facilitador?

41. ¿Qué te gustaría mejorar del equipo?

42. ¿Qué te gustaría mejorar del entorno?

43. Comentarios adicionales

9.3.3.2.4 Mi bitácora

A lo largo de todo este proceso voy a escribir un documento en el que refleje mi opinión sobre cómo se va desarrollando el proceso.

Este documento se organiza en semanas, aunque voy anotando cada pocos días los problemas y las experiencias reseñables que vamos (y voy) teniendo: conversaciones con la dirección, con mi equipo, con jefes de otros equipo, etc.

Aunque este documento no es valorable de forma objetiva, sí que puede funcionar como registro de las experiencias más reseñables que hemos tenido durante este período.

Espero que en general mis experiencias sean positivas al finalizar este proceso y esto quede reflejado en esta bitácora.

Quizás lo óptimo hubiera sido pedir esta opinión a cada uno de los miembros del equipo, como se hace con el estado de ánimo del equipo en los calendarios Niko-niko [SCHIFFER11], pero no les quería sobrecargar de trabajo y nuevas herramientas al principio, por lo que deseché esta opción.

10 Resultados de la primera evaluación del proceso: implantación de Kanban

10.1 Introducción

Antes de comenzar a mostrar resultados, vamos a recordar los proyectos con los que el equipo trabaja:

- **A:** Aplicación web con interfaz de servicios web de terceros (PHP + Python + Django).
- **B:** Aplicación web de gamificación (Python + Django).
- **C:** Gestión de eventos (PHP con *framework* propio).
- **D:** Mantenimiento de contenidos y maquetación de un portal para un cliente (OpenCMS).
- **E:** Maquetación de la web de nuestra empresa (HTML + CSS)
- **F:** Aplicación web de venta de entradas (Python + Django).
- **G:** Aplicación web de gestión turística (Python + Django).
- **H:** Adición de funcionalidad a una plataforma web de nuestro cliente para proporcionar interconexión con un tercero (PHP).
- **I:** portal con estadísticas, registros e interfaces de A (Python + Django).

Los proyectos B, D, E y H son proyectos en los que hay más de un desarrollador del equipo. El proyecto F tiene dos desarrolladores, uno en el equipo y otro fuera. Los proyectos A y C son proyectos en los que en la mayoría de los casos trabaja solo un único desarrollador. Esto hace que la encuesta de los proyectos A y C sea contestada por un desarrollador (por proyecto) y el resto se completen mediante un consenso entre todos los desarrolladores que trabajan sobre el proyecto.

De esta forma, tendremos una encuesta y por tanto un gráfico de profundidad por proyecto.

Por desgracia, los proyectos G, H e I no han implantado Kanban de una forma completa por lo que no tiene ningún sentido ver la profundidad de implantación.

Concretamente, el proyecto G no tiene un cliente que colabore con nosotros y por tanto, se usa un tablero de tareas, pero no hay esperanzas de poder mejorar el método mas allá. El cliente no conoce bien su modelo de negocio, no es claro a la hora de indicar requisitos y no revisa la funcionalidad como debiera. No estamos ante un cliente con el que se pueda trabajar con este método.

Por otro lado, el proyecto H es un proyecto atrasado sobre el que asumimos pérdidas desde el primer momento y que está próximo a su finalización. Por lo tanto, pensamos que no era adecuado cambiar la forma de trabajar tan próximo al final del desarrollo. Para colmo de males, la integración de nuestro código con el del cliente está siendo compleja debido a su alta deuda técnica. En

definitiva, un proyecto complicado con fuertes requisitos temporales para el que no podíamos cambiar la forma de trabajar.

Por último, el proyecto I es un proyecto muy retrasado, de poca prioridad y que en principio se basa en el proyecto A, ya que no es más que un portal de todas las operaciones que puede realizar el proyecto A. Dado que es un proyecto que acumula un sobrecoste de varios meses de trabajo y que no está siendo prioritario para el cliente, decidimos dejarlo fuera de la evaluación.

10.2 Resultados

10.2.1 Proyecto A

En este proyecto, el cliente es colaborativo, lo que se puede ver en que la visualización es de muy alta calidad. El tablero siempre refleja su estado al 100%. Además, el hecho de que el cliente tenga tanta implicación con el proyecto y haya aceptado el uso de Kanban¹¹ hace que el trabajo sea cómodo, agradable y haya un buen nivel de motivación entre los desarrolladores (sólo hay que ver el nivel de “Efectos”).

El principal problema es que consiste en el desarrollo de una interfaz web para muchos servicios web. Trabajar con servicios web de distintos proveedores, cada uno con su flujo concreto y con una documentación limitada, hace que la incertidumbre sea alta. Los obstáculos que aparecen son en muchas ocasiones nuevos (de ahí que el eje “Mejora” tenga ese valor de 5).

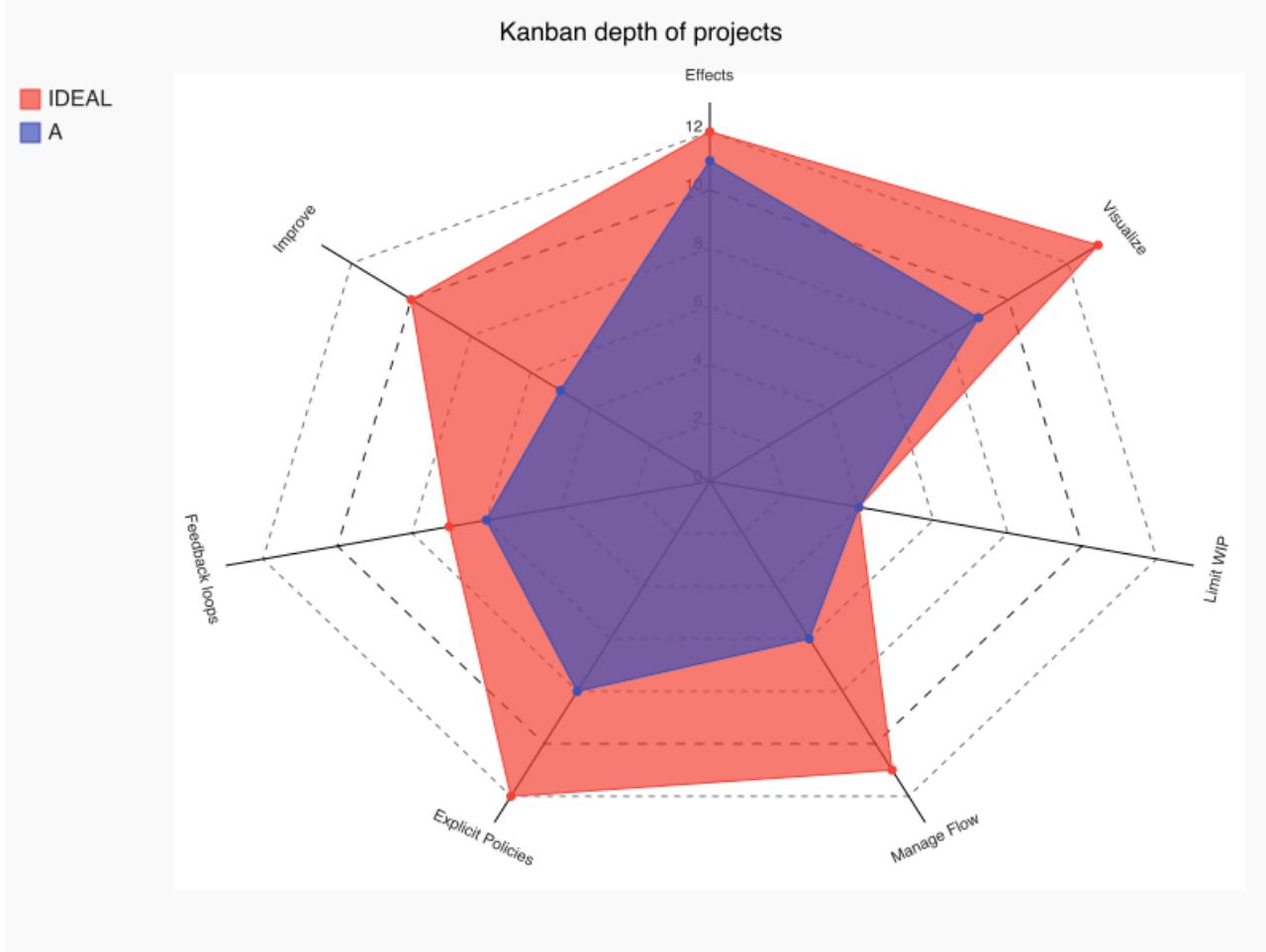
Hasta hace poco el objetivo del proyecto era un poco “difuso” y “cambiante”, de ahí que los valores de “Gestión de flujo” sean bajos.

En definitiva, los miembros del equipo creen que Kanban está mejorando la comunicación con el cliente y que todos los problemas se derivan de tener que depender de servicios externos mal documentados, con comportamientos poco descritos y con flujos de trabajo complejos de implementar.

En la siguiente gráfica radar se muestran en el fondo de color rojo los valores ideales y en el frente, los del proyecto A. Se puede ver como se tiene una buena profundidad de Kanban. Como decimos, se debe fundamentalmente al buen hacer del cliente que es metódico, ordenado e indica las incidencias lo más detalladamente posible y de la mejor forma posible.

Ni qué decir tiene que los miembros del equipo que trabajan con este cliente no quieren dejar el proyecto.

¹¹ Tal es así que, después de comprobar lo beneficiosa que ha sido la experiencia, el cliente ha comenzado a usar un tablero de tareas en su propio equipo de trabajo.



10.2.2 Proyecto B

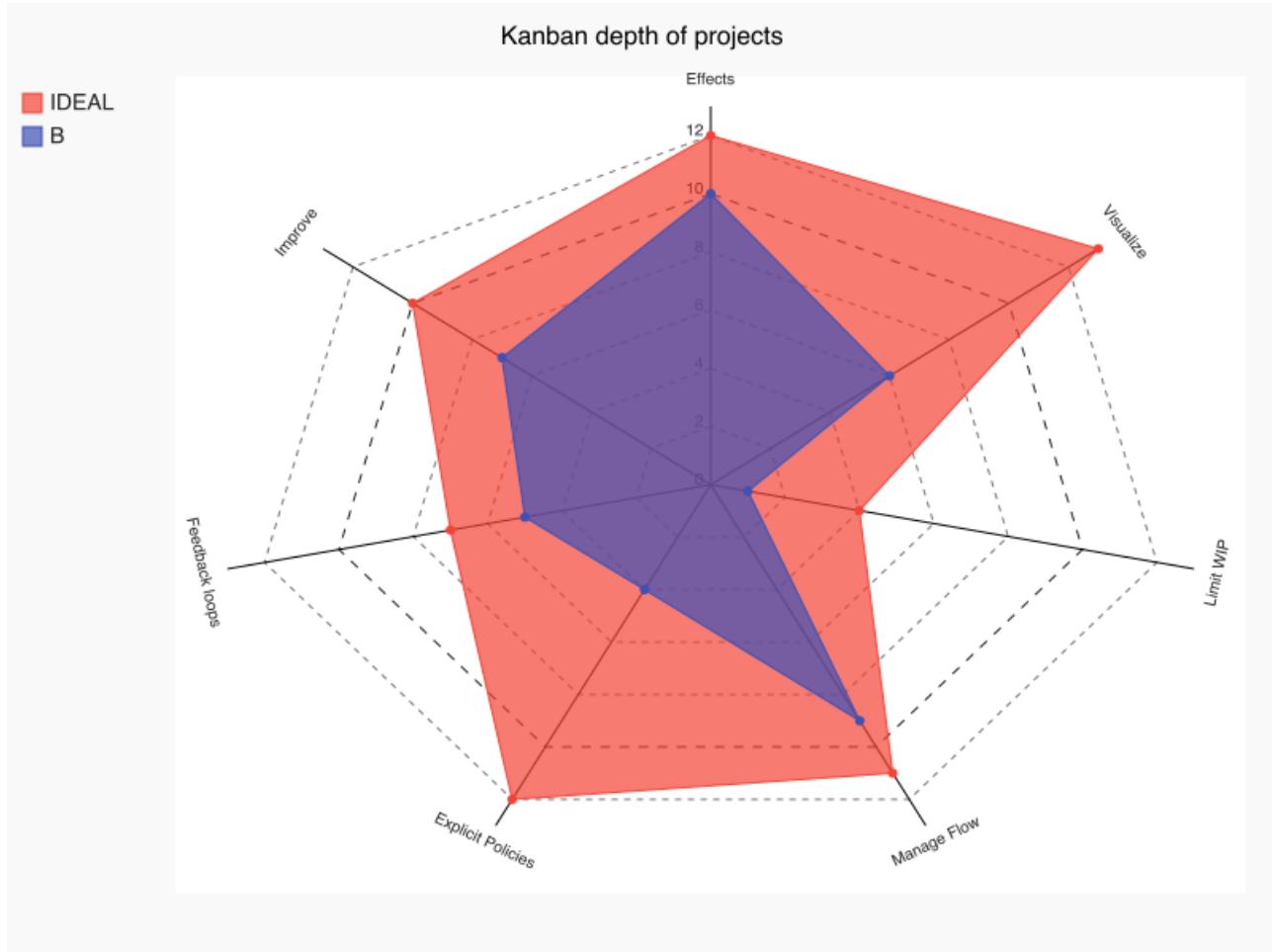
Este proyecto está muy bien detallado, pero el personal que lo está llevando desarrolla también el proyecto F y G. Esto hace que el proyecto tenga una velocidad de desarrollo muy baja. Las tareas tienen un tiempo *lead* muy alto y esto afecta a la motivación del equipo, que ve cómo el proyecto se alarga en el tiempo.

Es un proyecto con un buen cliente y con unas especificaciones muy bien definidas, pero la presión de los otros proyectos hace que sea muy complicado desarrollar Kanban y sacarle todo el partido. De hecho, no se han implantado políticas claras ni de límite de WIP ni de tipos de tareas. También, el estrés de los otros proyectos hace que los desarrolladores no trabajen según lo indicado en el tablero y no lo actualicen hasta después de que se terminen las tareas.

En este proyecto se han hecho algunas reuniones con el cliente para mostrárselo, pero dada la velocidad que se lleva, no se pueden ni hacer reuniones mensuales. Se hizo una reunión y el cliente aportó mucha retroalimentación, pero sólo se ha hecho una.

Se podría decir que este proyecto está casi en *standby*. De hecho, se espera que cuando se terminen con los proyectos F y G, se pueda trabajar en este proyecto y, por fin, desarrollar Kanban como

debe hacerse.



10.2.3 Proyecto C

El proyecto C es un proyecto heredado con tiene multitud de clientes.

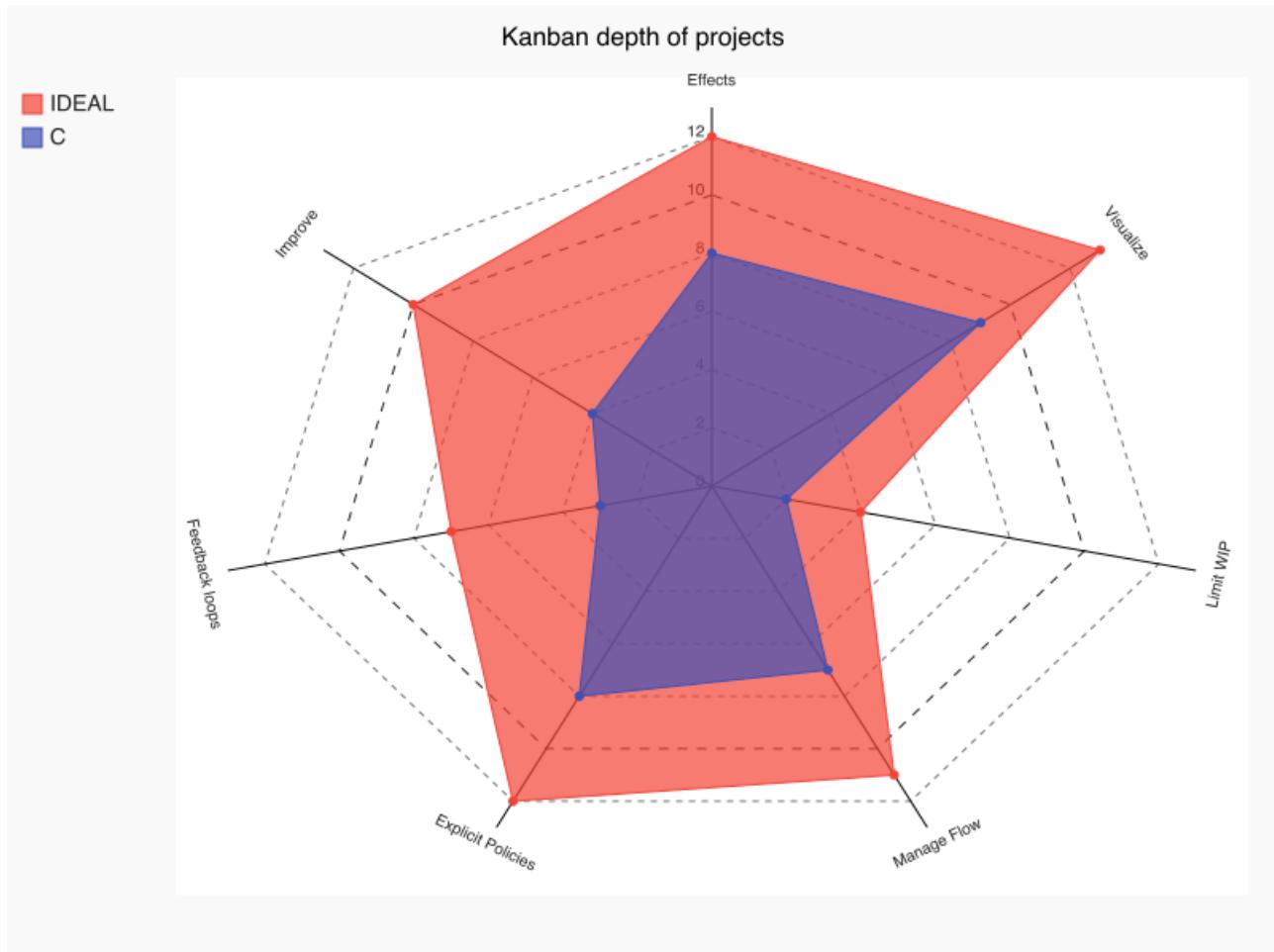
Funciona en la mayoría de los casos correctamente, pero tiene una alta deuda técnica, lo que lleva a la desmotivación al equipo al no poder realizar todas las mejoras (refactorizaciones) que quisieran.

También, no hay documentación exhaustiva, lo que es frustrante a la hora de corregir un error, porque no se conoce el comportamiento correcto. Muchas veces hemos de guiarnos por lo que dice el equipo comercial.

En este proyecto hay colaboración con el equipo comercial (que actúa como *product owner*) y hay que trabajar más para tener mejores flujos de trabajo y un mejor tratamiento de las incidencias, cuando hay incidencias críticas, éstas ni siquiera se comunican por el tablero Kanban.

El equipo comercial hace lo que pueda, pero en un proyecto de estas características, a veces es mejor rehacerlo de 0, o si acaso, invertir en una refactorización de todo el código.

En este caso, se puede ver cómo tener un proyecto con alta deuda técnica se comporta como un sumidero de energía, motivación y tiempo, afectando a la capacidad del equipo de implantar una técnica de trabajo tan potente como ésta.



10.2.4 Proyecto D

Este proyecto se basa en gestionar una instalación de OpenCMS y no requiere desarrollo software.

La dificultad de este proyecto radica en la poca colaboración y responsabilidad del cliente. Revisiones que no se hacen, documentación entregada a última hora, cambios sobre cambios sobre más cambios que deberían haber sido validados...

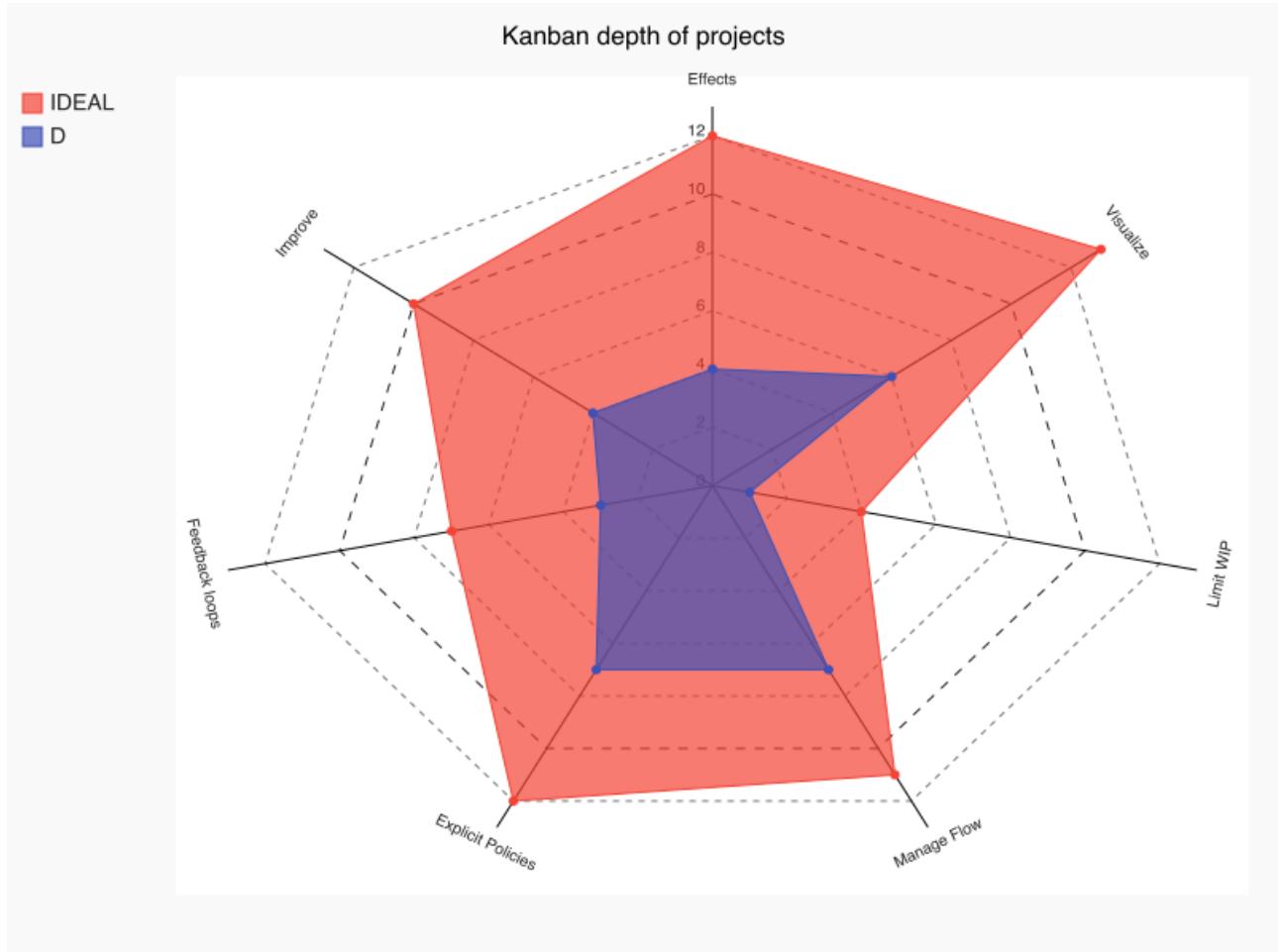
Es decir, existe un flujo de trabajo y listas de comprobación de distintos tipos de tarea, pero el cliente no hace su trabajo y en fechas próximas a las entregas es cuando lo hace, lo que dificulta la planificación del equipo.

Colaboramos con otro equipo que es el que nos hace las tareas de comunicación con el cliente final. Para colmo, ellos nos interrumpen sin problema ante errores o incidencias críticas, lo que desmotiva

y estresa al equipo.

Las tareas son sencillas pero, al haber multitud de incidencias inesperadas, todo se complica de sobremanera. Es sorprendente cómo se muestra esto en el diagrama de profundidad de Kanban. Básicamente el tablero muestra las tareas, pero no es definitivo de ninguna forma.

Es un proyecto que está suponiendo una carga a algunos miembros de mi equipo que desean progresar en su carrera profesional y no gestionar contenidos de una plataforma web.



10.2.5 Proyecto E

Este proyecto es una maquetación web de la página de la empresa. Es un proyecto perfecto para hacer uso de Kanban ya que son tareas sin complejidad técnica y de pequeño tamaño.

El departamento de comunicación está haciendo un buen trabajo en la creación de tareas, gestión de flujos y limitación del WIP. Son tareas de como máximo 4 horas y todas basadas en la modificación de código HTML y Javascript, ni siquiera hay tareas de desarrollo de funcionalidad. Entonces ¿por qué tiene estos valores tan bajos en algunas de las dimensiones?

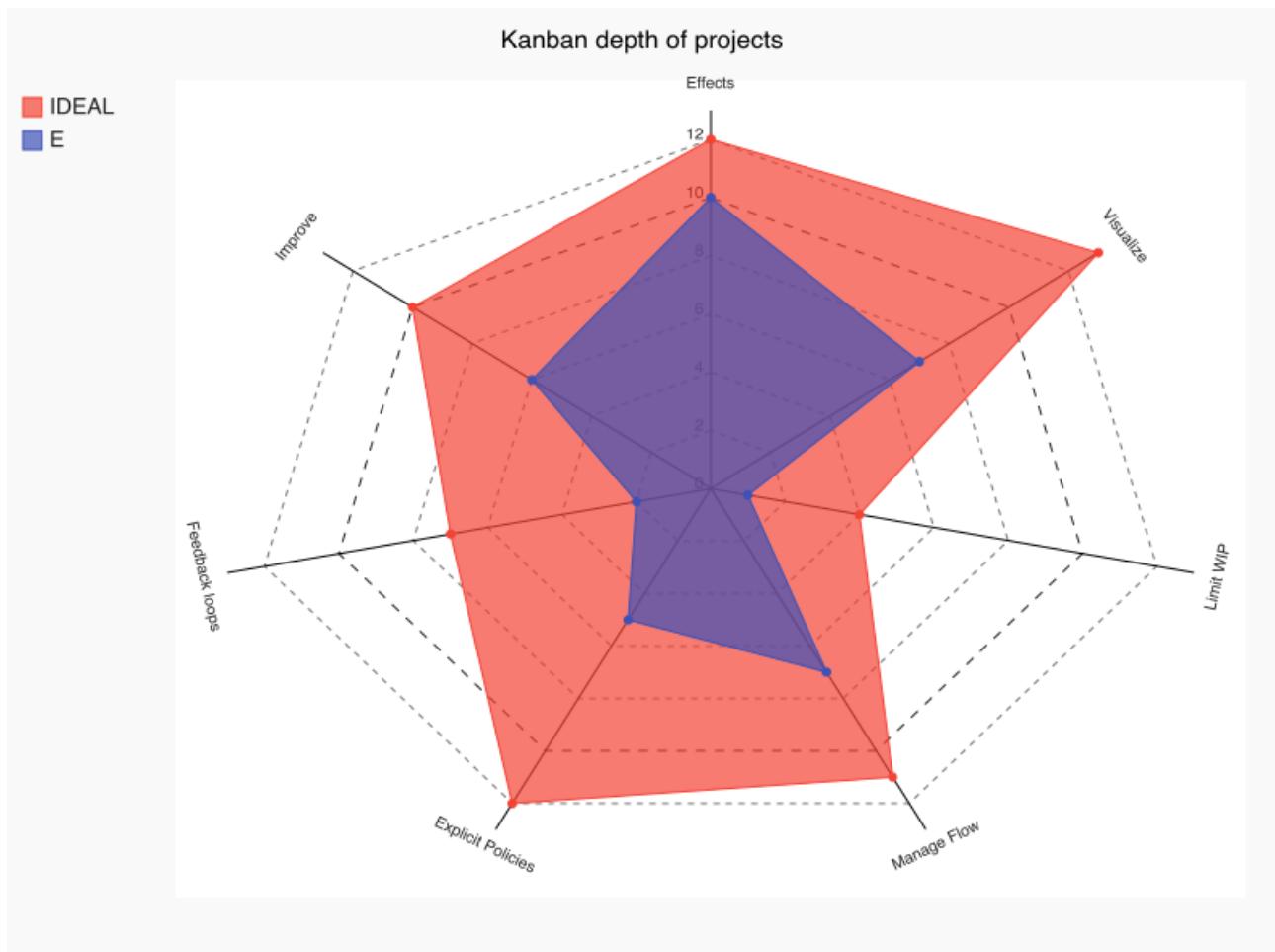
Por la falta de experiencia y formación del desarrollador principal. Le cuesta entender el flujo de

trabajo. Yo mismo he tenido que deshacer algunos de sus cambios porque introducían regresiones en la web. Si no se respeta el límite de WIP es por la gran cantidad de tareas que he tenido que devolver a la columna de “En desarrollo” desde la de “En revisión”.

No hay políticas ni proceso de estimación, pero también es cierto que el tipo de tareas no son complejas como para no poder ser estimadas por un maquetador con experiencia. Las políticas que hay son suficientes, pero hay que seguirlas.

Esto nos sirve como enseñanza de que más allá del proceso, hay que ser exigente con el trabajo, seguir buenas prácticas de desarrollo con revisiones y siendo exigente.

También, en este caso no hay jerarquía de tareas ni tipos de tareas, lo que ha supuesto un valor bajo a “Visualización”.



10.2.6 Proyecto F

Este proyecto está terminado y tiene una buena calidad. Funciona bien y en la mayor parte de los casos sólo hacemos labores de soporte técnico y pequeñas mejoras.

Yo personalmente trabajo en este proyecto con un compañero de otro equipo y ambos tenemos una

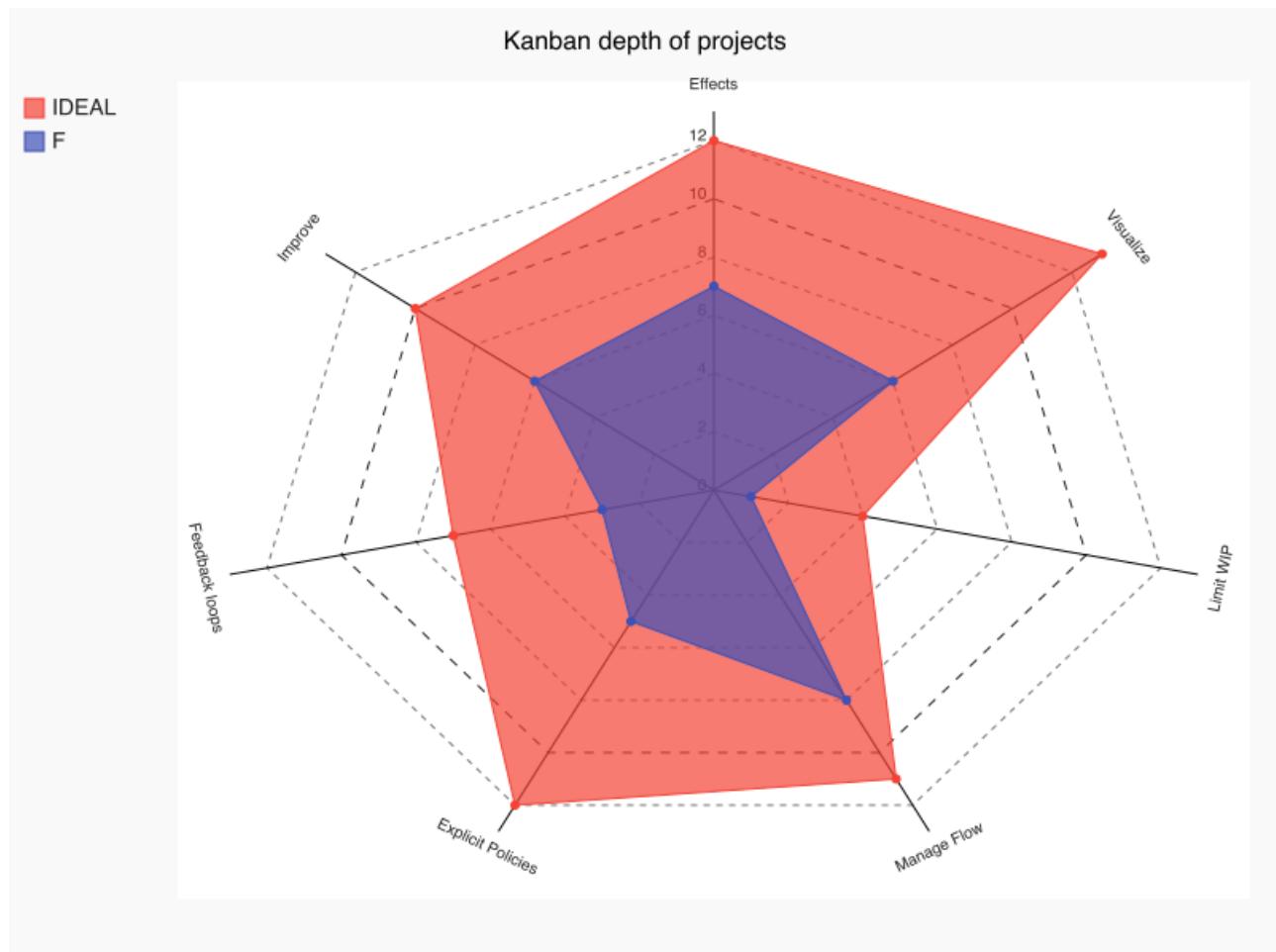
buena y estrecha relación.

Este proyecto tiene una complejidad comercial en el sentido de que cada cliente tiene un flujo de trabajo distinto y quiere que la aplicación web se adapte al suyo. Esto se refleja en nosotros respondiendo dudas comerciales sobre cómo hacerlas

El tablero de tareas no tiene un gran movimiento más allá de pequeñas mejoras que ocurren con una frecuencia baja. ¿Entonces, refleja el estado del proyecto? Sí y no, las tareas de soporte no existen en el tablero pero ocupan tiempo, lo que hace que en una gran parte, el tablero no sea real.

Estas tareas de soporte, de modelado de las necesidades de los clientes se han de explicar constantemente al equipo comercial (que no tiene conocimientos técnicos de ninguna clase). Las interrupciones son frecuentes y son un gasto que no se contempla.

En definitiva, desde el punto de vista del desarrollo exclusivamente, Kanban está implantado con una profundidad adecuada para el poco tiempo que lleva. El conocimiento y calidad de los desarrolladores hace que el desarrollo no sea el problema. El problema es que el trabajo de desarrollo es la menor parte en este proyecto.



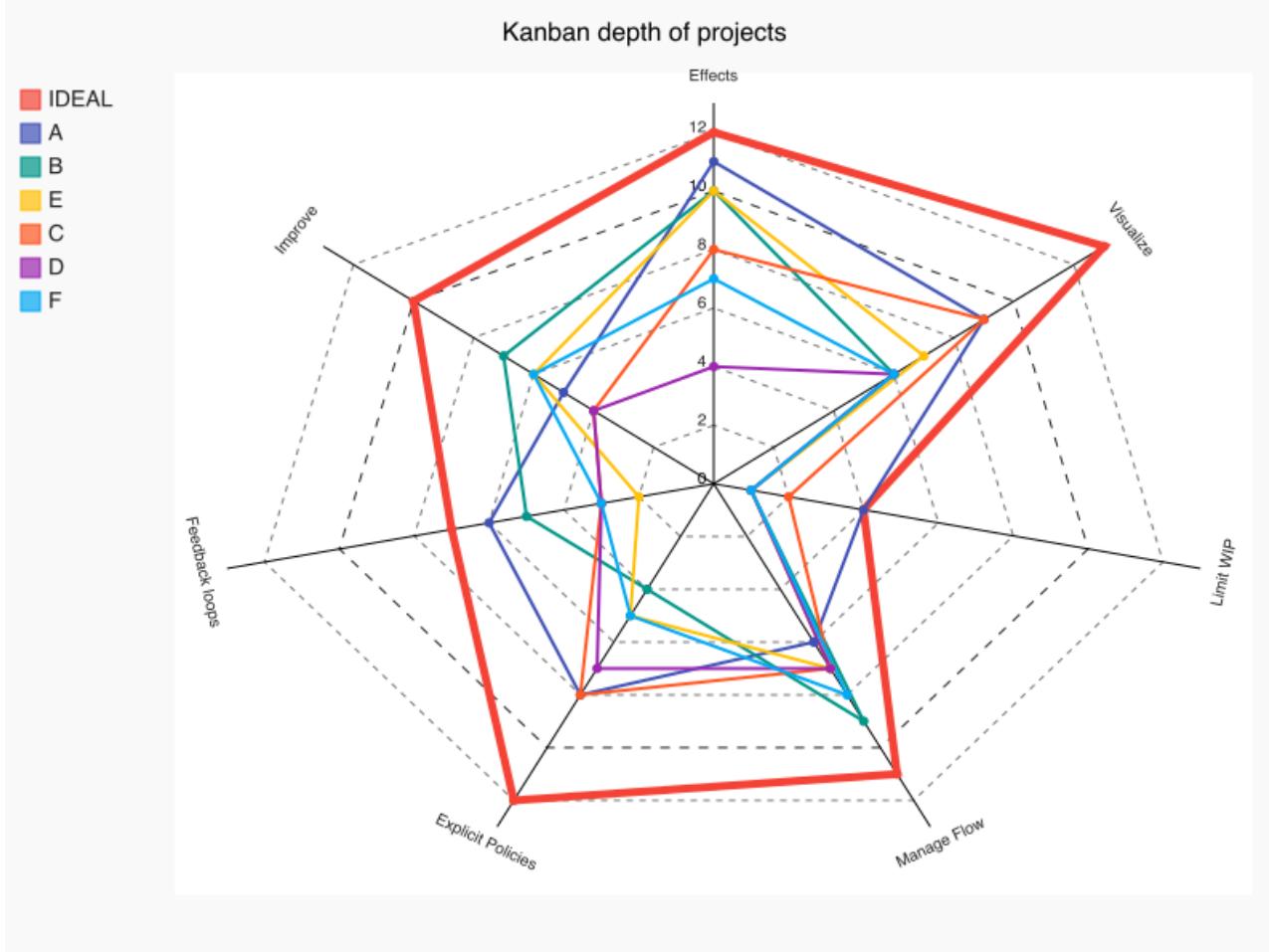
10.3 Conclusiones

En la mayoría de las veces no dependemos sólo de nosotros a la hora de implantar de forma completa un marco de trabajo basado en Kanban.

Hemos mostrado que la causa de una mala implantación de Kanban puede ser:

- **El cliente:** su colaboración es fundamental.
- **El proyecto:** hay proyectos con una complejidad intrínseca debido a la comunicación con otras terceras partes.
- **El equipo:** si el equipo no es profesional y retrasa el trabajo esto afecta a la moral del resto y del cliente.
- **El resto de equipos:** si se trabaja con otros equipo, bien de desarrollo o comerciales, es necesario que trabajen respetando las normas y siguiendo las recomendaciones de este método de trabajo.
- **El resto de proyectos:** hemos hablado de que el proyecto B no está recibiendo atención y no se están desarrollando sus tareas a un ritmo estable. Deberemos en un futuro implementar algún sistema para evitar la inanición de este proyecto.
- **Los desarrolladores:** evidentemente esta evaluación es subjetiva puesto que cada desarrollador será más o menos exigente. Además, el hecho de que los proyectos tengan pocos desarrolladores hace que las opiniones estén muy sesgadas.

Aquí podemos ver la variabilidad tan grande que tienen los proyectos que trabajamos:



En definitiva, esta evaluación tiene como objetivo servir de punto de partida para las siguientes. En seis meses volveré a realizar una evaluación de cada uno de los proyectos y a comparar los resultados con los aquí obtenidos y anteriormente mostrados.

Lo primero que voy a tratar de hacer en estos seis meses es reducir la subjetividad de la prueba. A día de hoy, sólo uno (o si acaso, un par) de desarrolladores llevan cada proyecto. Mi objetivo es hacer que todos los desarrolladores participen en todos los proyectos y por tanto la evaluación se haga por un consenso. Esto suavizará las opiniones más extremas de los desarrolladores y nos hará obtener un resultado menos dependiente de la opinión de desarrolladores concretos. El objetivo es tener un equipo unido, también en cuanto a la valoración y opinión de los proyectos en los que trabaja.

Luego trataremos de mejorar nuestra forma de trabajar con el equipo comercial, que parece ser una de las causas de la generación de muchos tiempos muertos y de baja implantación de Kanban. En especial en cuanto a la poca capacidad de previsión de éstos en cuanto a volumen de trabajo, creando picos que, interrumpen el trabajo en otros proyectos.

El resto de proyectos requerirán un estudio particular para cada uno de ellos.

11 Resultados de la segunda evaluación del proceso: cuestionarios de evaluación del equipo y del proceso

El día 4 de Mayo del 2016 se le pasó a la dirección, a un dueño del producto de dos proyectos y al equipo de desarrollo unos cuestionarios con la intención de que contestasen de la forma más sincera a las preguntas.

Se les indicó que no habría “respuestas correctas” y que realmente era necesaria su colaboración, tanto por el bien del equipo como de la empresa.

A continuación, se muestran los resultados de las encuestas para cada uno de los implicados:

11.1 Dirección

El cuestionario se pasó al director ejecutivo (CEO) y al director de operaciones (COO).

A continuación mostramos las preguntas numéricas

11.1.1 Preguntas numéricas

Cuestión	CEO	COO	Media
1 Estoy contento con la cultura del equipo	3	4	3,42
2 Veo una mejora en la productividad del equipo	4	5	4,44
3 Veo mayor motivación en los miembros del equipo	3	4	3,43
4 El equipo me proporciona métricas adecuadas	5	5	5
5 El equipo me da los informes en el formato adecuado	5	4	4,44
6 La calidad en los proyectos del equipo ha mejorado	4	5	4,44
7 El equipo aporta valor a la empresa	4	5	4,44
8 El equipo aporta valor a los clientes	3	5	3,75
9 El software generado es de suficiente calidad	3	5	3,75
10 El equipo genera trabajo de forma predecible	4	N/A	4
11 Los clientes están satisfechos con este marco de trabajo	4	N/A	4
Media	3,69	4,66	4,05

11.1.2 Preguntas abiertas

11.1.2.1 ¿Qué te gustaría mejorar del manager?

Las preguntas sobre el manager el director ejecutivo le indicó a éste la necesidad de potenciar el buen ambiente en su equipo con ciertos momentos lúdicos. Esta idea creo que también se refleja en la valoración a la cuestión 1, ya que el CEO no valora de forma totalmente positiva la cultura que

estoy implantando. Quizás demasiado nivel de exigencia y falta de quizás un aprendizaje lúdico (también llamado *gamificación*).

El COO no contestó.

11.1.2.2 ¿Qué te gustaría mejorar del equipo?

El CEO prefiere esperar a que transcurra más tiempo. Lógico. Empezamos el proceso a mediados de Febrero. Son necesarios mínimo seis meses para poder valorar un cambio de esta naturaleza de forma objetiva.

El COO me indicó que uno de los miembros del equipo no está lo suficiente motivado. Ha estado trabajando mucho tiempo solo en su proyecto, a partir de Junio me pondré a trabajar personalmente con él.

11.1.2.3 Comentarios adicionales

El CEO está contento del proceso de innovación y del esfuerzo que estoy poniendo en llevarlos a cabo. Me hace explícito su apoyo lo cual indica que uno de los pilares fundamentales de todo proceso de cambio lo he conseguido.

El COO me da la enhorabuena y habla de que “estoy haciendo un buen trabajo”.

11.1.3 Análisis sobre los resultados

Los valores son medianamente elevados. El COO ha sido más benévolos que el CEO pero creo que ambos valoran positivamente este proceso.

Es destacable el hecho de que los informes quincenales que envío al COO parecen ser adecuados y el CEO también está de acuerdo con ello.

Por otro lado, también me llama la atención cómo hay discrepancia en cuanto a si el equipo hace software de calidad y si aporta valor a los clientes.

Quizás la opinión del COO es más reseñable por el hecho de estar más próximo a nosotros ya que hace labores de planificación y seguimiento de proyectos, estando al tanto del estado de cada proyecto y de las relaciones con los clientes.

Así, dado que el CEO como el COO opinan de forma positiva en la mayoría de las preguntas, parece que estamos obteniendo resultados positivos.

Parece que el CEO quiere un cambio de actitud en el sentido de que es una persona que no está contento con un equipo de desarrollo en el que no haya momentos de descanso, bromas y demás “artefactos” para mejorar el buen ambiente.

El COO es más práctico y me indica que uno de los desarrolladores tiene que mejorar a nivel de motivación y a calidad de trabajo. Lleva más de un año y le falta guía en cuanto a su trabajo.

11.2 Dueño del producto

En nuestra empresa el dueño de producto para dos de nuestros proyectos es el director comercial por lo que le entregamos el cuestionario el día 6 de Mayo y nos lo devolvió al día siguiente con los siguientes resultados:

11.2.1 Preguntas numéricas

Cuestión	Director comercial
1 El equipo desarrolla la funcionalidad en el orden adecuado	5
2 El equipo aporta valor más allá de desarrollar sólo lo que le indico	4
3 Veo al equipo comprometido con el proyecto	3
4 La herramienta que usamos es adecuada para comunicarme con el equipo	3
5 La herramienta que usamos es adecuada para visualizar el estado del proyecto	4
6 Las entregas no se retrasan	4
7 El tiempo de respuesta ante las incidencias es el adecuado	5
8 El software desarrollado tiene calidad	3
9 El equipo recibe retroalimentación de forma continua bien con nuevas tareas o modificaciones de tareas	5
10 El equipo hace caso a la retroalimentación que le doy	5
11 El software se adapta bien a mis necesidades	4
12 Hay buena comunicación con el equipo	4
Media	3,69

11.2.2 Preguntas abiertas

11.2.2.1 ¿Qué te gustaría mejorar del equipo?

El director comercial habla de que el facilitador ha hecho un “salto cualitativo muy grande” pero de que la mejora ha de ser constante.

También hace mención a que la relación con el equipo no ha de ser tan jerárquica y permitir casos concretos en los que los protocolos no se sigan al 100%.

11.2.2.2 Comentarios adicionales

Aquí habla de que el proceso de mejora hace que sea un ejemplo a seguir en el resto de la empresa.

11.2.3 Análisis sobre los resultados

Parece que desde el punto de vista del dueño del producto falta principalmente compromiso y quizás buscar una herramienta que proporcione mayor grado de flexibilidad en la comunicación. La falta de compromiso no sé a qué se debe y no casa con el que haya retroalimentación constante por nuestra parte. La herramienta es la que es, para tener más flexibilidad deberíamos soportar más

interrupciones y no estoy dispuesto a ceder en ese punto (salvo incidencias críticas). Entiendo que desde su punto de vista, la herramienta es muy limitada, pero no podemos perder tiempo de desarrollo y menos con la cantidad de proyectos que tenemos.

También, desde su punto de vista, no ha habido un incremento de calidad en el software. Esto puede explicarse en el sentido de que, en el momento de la encuesta, no hemos desarrollando interfaces aún con el protocolo indicado en la sección de usabilidad. Por eso, la “calidad” que él percibe es la misma que siempre. Evidentemente él no va percibir la calidad en el código, sólo en las interfaces.

Si bien sienta bien el reconocimiento personal, hubiera preferido más información sobre su opinión del equipo. El hecho de que considere el equipo una jerarquía, estando yo en la cúspide y el resto del equipo debajo no me gusta. Es obvio que no estoy haciendo todo lo bien que podría mi trabajo de facilitador. Quizás confunde el proceso de guía con el hecho de que sea yo quien tome todas las decisiones en mi equipo (lo cual no es cierto). Pondré en común estos puntos al equipo y espero que me den una buena retroalimentación.

11.3 Equipo

A cada uno de los miembros del equipo se les presentó el cuestionario del que se muestran a continuación sus resultados:

11.3.1 Preguntas numéricas

Cuestiones	1	2	3	4	Media
1. La formación que he recibido sobre Kanban es la adecuada	3	N/A	4	4	3,6
2. Las herramientas son las apropiadas para implantar Kanban	5	N/A	4	4	4,29
3. La transición de la forma de trabajar se ha hecho de forma adecuada	4	4	5	4	4,21
4. Creo que podemos mejorar nuestro rendimiento aún más	4	4	5	3	3,87
5. He visto mejora en mi trabajo desde que comenzó el proceso	4	5	5	4	4,44
6. La arquitectura software es más adecuada a cada proyecto	2	3	4	4	3
7. Nuestro software es de mejor calidad	3	4	4	4	3,69
8. Hacemos código limpio	4	4	4	4	4
9. Nuestro software es más usable	3	4	3	3	3,2
10. Todo el software que hacemos está documentado	2	4	3	3	2,82
11. Hago uso de bocetos siempre que tengo que explicar una interfaz compleja	3	3	3	5	3,33
12. El equipo colabora de forma eficiente	4	5	5	5	4,7
13. El equipo ha mejorado su productividad desde que se implantó este marco de trabajo	4	4	5	4	4,21
14. Estoy viendo que mi trabajo tiene relevancia en la empresa	4	5	4	3	3,87
15. No hay picos de trabajo y el ritmo es el adecuado	2	3	4	2	2,53
16. Los cuellos de botella no son responsabilidad de nuestro equipo	4	5	5	3	4,07
17. Nos organizamos a nosotros mismos como estimamos oportuno	5	4	4	4	4,21
18. Creo que se puede visualizar en cada momento el estado del proyecto fácilmente	4	4	5	4	4,21
19. Mis compañeros de equipo me motivan	5	4	5	5	4,71
20. Mi motivación ha crecido desde que ha comenzado este proceso	5	4	5	4	4,44
21. Confío en mis compañeros de equipo para terminar proyectos	5	5	5	5	5
22. Siento que se me escucha en el equipo	5	5	5	5	5
23. Estoy contento de formar parte de este equipo	5	5	5	5	5
24. Todos los miembros del equipo contribuyen en el trabajo	5	5	5	5	5
25. No tengo interrupciones que me hacen perder tiempo en el día a día	4	5	1	3	2,24
26. El entorno es el adecuado para desarrollar (ruido, luz, comodidad...)	3	3	2	3	2,67
27. La falta de documentación nos está suponiendo un problema en algunos proyectos	3	2	5	5	3,24
28. Los clientes están colaborando activamente en el proceso de mejora	1	3	3	5	2,14
29. Otros equipos están colaborando activamente en el proceso de mejora	3	3	3	5	3,33
30. El método de cajas de tiempo mejora la productividad	4	4	4	3	3,69
31. Tenemos los recursos suficientes para seguir con el proceso de implantación	4	4	N/A	3	3,6
32. Compañeros de otros equipos me motivan con su ejemplo	5	3	3	5	3,75
33. Las reuniones diarias son adecuadas	3	4	4	4	3,69
34. He aprendido de las reuniones diarias	5	4	5	5	4,71
35. Las reuniones de retrospectiva son adecuadas	5	4	5	4	4,44
36. He aprendido de las reuniones de retrospectiva	5	4	5	5	4,71
37. Mi nivel de estrés se ha reducido gracias a este nuevo marco de trabajo	3	3	4	3	3,2
38. Siento que el facilitador me escucha	5	5	5	5	5
39. No volvería a la forma de trabajar anterior	5	5	5	5	5
Media	3,45	3,85	3,78	3,87	3,73

11.3.2 Preguntas abiertas

11.3.2.1 ¿Qué te gustaría mejorar del facilitador?

El desarrollador 1 no contesta.

El desarrollador 2 no contesta.

El desarrollador 3 indica la necesidad de gestionar mejor el tiempo de las reuniones.

11.3.2.2 ¿Qué te gustaría mejorar del equipo?

El desarrollador 1 no contesta.

El desarrollador 2 hace mención a que hay muchos proyectos no son compartidos.

El desarrollador 3 no contesta.

11.3.2.3 ¿Qué te gustaría mejorar del entorno?

El desarrollador 1 indica que el ruido de la oficina sigue siendo un problema en algunas horas a lo largo del día.

El desarrollador 2 también hace mención al ruido que hay en la oficina en determinados momentos.

El desarrollador 3 también hace constar que el ruido y las interrupciones de miembros de otros son elementos que empeoran su trabajo.

11.3.2.4 Comentarios adicionales

El desarrollador 1 hace mención a que las reuniones diarias son demasiado largas.

El desarrollador 2 no contesta.

El desarrollador 3 no contesta.

11.3.3 Análisis sobre los resultados

Lo primero, con respecto al marco de trabajo y al proceso de mejora del desarrollo software, vemos cómo todos los miembros valoran de forma positiva la transición. Todos creen que tanto la calidad como el rendimiento del equipo. Es cierto que algún miembro es escéptico en cuanto a si puede mejorar su forma de trabajar aún más, pero el resto sí que lo creen.

La usabilidad no se ha visto mejorada de forma sustancial dado que no hemos tenido tiempo de generar nuevos bocetos de interfaces porque las incidencias y cambios que estamos recibiendo no tienen interfaces de usuarios. Cuando les he pedido que usen bocetos y diagramas (antes de la implantación de la mejora en usabilidad) hemos recibido retroalimentación positiva del cliente y de otros equipos.

La documentación del software es un tema que hemos de mejorar, me preocupa el hecho de que casi todo el equipo conteste que no ha habido mejora e incluso un miembro hable de que no está de

acuerdo con hacer documentación. También el hecho de tener proyectos heredados nos hace mucho daño. Uno de los proyectos más grandes que tenemos no tiene documentación: no ya en un fichero de texto, sino siquiera en el código. Es un tema que tendré que tratar con ellos.

Kanban no está solucionando los problemas de control en el flujo de trabajo y eliminación de “picos de trabajo”. Esto se debe, en primer lugar, a los proyectos heredados. Los desarrolladores que tienen proyectos heredados se quejan de este particular, los que no, no. Pero no tenemos recursos para refactorizar completamente los proyectos heredados, por lo que hasta que no pase un tiempo no veremos mejoras en estos proyectos.

Por otro lado, también estoy viendo cómo es necesario un enfoque de flujo constante a nivel de organización al completo. Si los dueños del producto no nos proporcionan un flujo constante de trabajo, sino que éste es esporádico, no podremos usar Kanban de forma correcta. Tiene que haber un flujo constante de tareas y éste debe estar sincronizado con él en función del *lead* del tablero.

Esto está relacionado con que nuestros clientes (internos y externos) no colaboran de la misma forma. Por eso tenemos proyectos que tienen clientes que se vuelcan con el proyecto y trabajan siguiendo el proceso, mientras otros directamente sólo se comunican con nosotros por correo electrónico y evitan cualquier tarea adicional.

Ahora mismo los clientes más motivados con este método son los dueños de producto internos, esto es, parte del equipo comercial.

Volviendo a los resultados del cuestionario, el tablero Kanban sí que permite una visualización real del estado de los proyectos. Esto es vital considero que es un paso fundamental para usar Kanban.

Las reuniones están teniendo un buen efecto en todos los desarrolladores y todos manifiestan que están aprendiendo y están creciendo como desarrolladores. El hecho de que un desarrollador indique que las reuniones diarias no son adecuadas lo clarifica en las preguntas abiertas indicando que las reuniones diarias “durán demasiado”.

En definitiva, con respecto al método, el equipo está contento con el sistema, pero no indican una reducción del nivel de estrés sustancial

Luego, con respecto a la motivación, vemos como el equipo está alcanzando bastante confianza en sí mismo y se siente respaldado por el resto de miembros y el facilitador. Hay buen ambiente y trabajo en equipo, eso es muy importante. Un entorno humano que no tiene calidad no puede hacer que surja un buen equipo de trabajo.

La parte del entorno que queda fuera de nuestro control sigue siendo un problema. En especial las interrupciones y el ruido (como luego contestarán varios desarrolladores). De ahí que no haya satisfacción en este tema. Las interrupciones han bajado con respecto a veces anteriores, pero el ruido sigue siendo elevado, sobre todo el proveniente de la zona comercial debido a teléfono, reuniones con clientes y, por supuesto, al día a día.

Como conclusión final, creo que nos estamos quedando cortos en la implantación debido a la

complejidad de los proyectos heredados, a no tener varios desarrolladores por proyecto y a la ausencia de una documentación de calidad para cada proyecto. Pero, vamos en el buen camino. El equipo se siente respaldado y con un nivel de satisfacción elevando. No sólo con el método, sino con el entorno. Tenemos una sensación de que estamos haciendo un trabajo de mejor calidad y poco a poco eso se está haciendo patente en nuestra forma de trabajar y en nuestros proyectos. Hay que retocar algunos detalles del método y potenciar que los dueños de producto tengan una mejor planificación a la hora de solicitar cambios, pero en su conjunto, la implantación está obteniendo resultados buenos.

12 Evaluación de las métricas de Kanban por proyecto

Lo primero es recordar otra vez los proyectos sobre los que estamos trabajando:

- **A:** Aplicación web con interfaz de servicios web de terceros (PHP + Python + Django).
- **B:** Aplicación web de gamificación (Python + Django).
- **C:** Gestión de eventos (PHP con *framework* propio).
- **D:** Mantenimiento de contenidos y maquetación de un portal para un cliente (OpenCMS).
- **E:** Maquetación de la web de nuestra empresa (HTML + CSS)
- **F:** Aplicación web de venta de entradas (Python + Django).
- **G:** Aplicación web de gestión turística (Python + Django).
- **H:** Adición de funcionalidad a una plataforma web de nuestro cliente para proporcionar interconexión con un tercero (PHP).
- **I:** portal con estadísticas, registros e interfaces de A (Python + Django).

Los proyectos A, B e I son los únicos que tienen desarrollo puro. El resto de proyectos tienen desarrollos y mantenimientos.

Los proyectos G, H e I no han usado el método Kanban por distintos motivos que recuerdo a continuación.

El proyecto G no tiene colaboración del cliente, que simplemente solicita cambios por correo. Trabajamos con base en prototipos que enseñamos al cliente y éste nos indica

Los proyectos H e I son proyectos retrasados que van a alcanzar el doble de lo presupuestado así que tratamos todo su tiempo de desarrollo como *desperdicio*. El objetivo es terminarlos cuanto antes, por lo que no podemos permitirnos el lujo de hacer “experimentos” con ellos.

Para cada proyecto mostramos las siguientes métricas:

- Tiempo medio de las tareas en cada estado.
- *Lead time*¹²
- *Cycle time*¹³.
- Estados que generan cuellos de botella.
- Desarrolladores que generan cuellos de botella¹⁴.

¹² Recordemos, el tiempo que tarda la tarea en desarrollarse, esto es, tiempo que pasa en el tablero.

¹³ Tiempo de desarrollo puro. O lo que es igual, el tiempo desde que comienza en el desarrollo hasta que se entrega al cliente.

¹⁴ Recordemos que por lo general en nuestro equipo cada tarea es realizada por un único desarrollador, así que si en el

Las mediciones se tomaron el día 8 de Mayo. Esto se debe a que la mayoría de los proyectos de mantenimiento estaban entrando en un estado de parálisis y sólo estábamos con tareas de proyectos de desarrollo y refactorización. Así, parecía que no tenía sentido esperar otra semana en obtener los datos porque, a efectos prácticos no iban a cambiar.

A continuación vamos a mostrar las métricas de cada uno de los proyectos y el análisis de la situación en la que se encuentra cada uno de ellos:

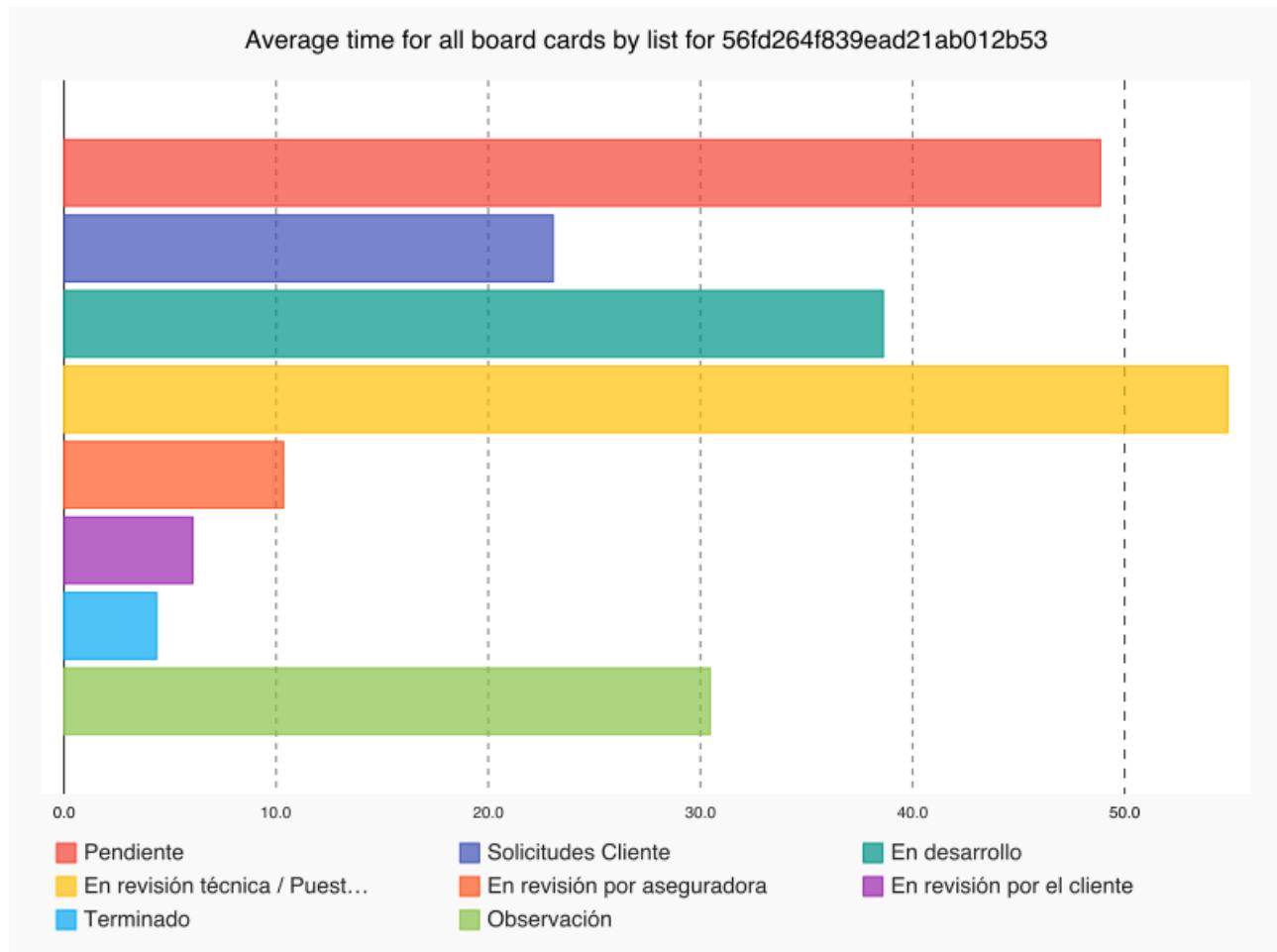
12.1 Proyecto A

Recordemos que este proyecto es un proyecto de desarrollo de interfaces web para servicios de terceros. Tiene una gran deuda técnica y la complejidad mayor reside en la comunicación con los servicios de terceros en el que cada uno de estos tienen su propio protocolo para realizar las mismas operaciones.

Vamos a mostrar las mediciones para este proyecto:

12.1.1 Mediciones

12.1.1.1 Tiempo medio de las tareas en cada estado



momento de la revisión tiene que moverse un estado atrás, en principio es responsabilidad de dicho desarrollador.

12.1.1.2 Lead

Media: 101.18 h.

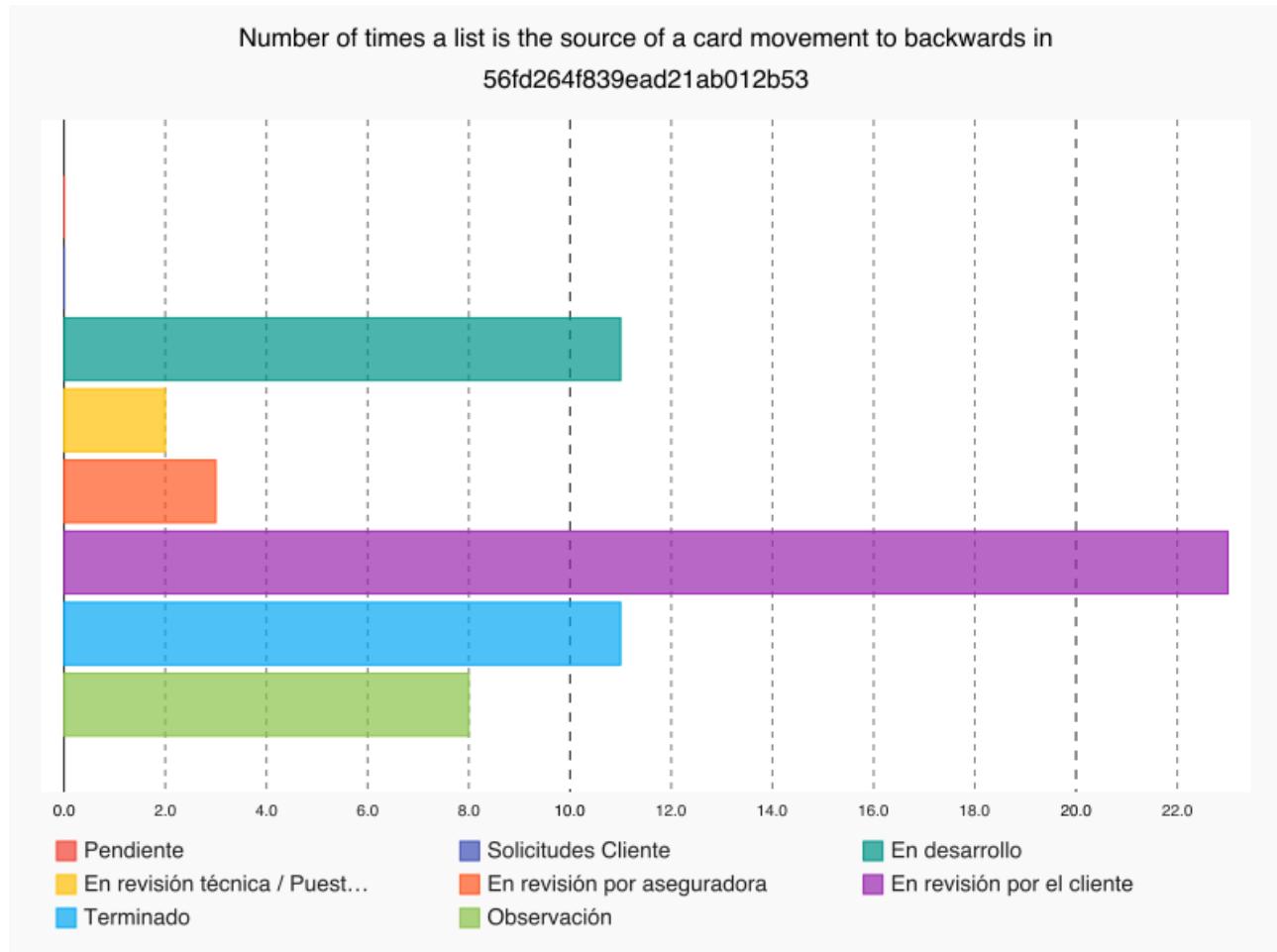
Desviación estándar: 145.06

12.1.1.3 Cycle

Media: 91.00 horas.

Desviación estándar: 140.18 horas.

12.1.1.4 Estados que ocasionan vueltas atrás



12.1.1.5 Tiempos de desarrollo

El desarrollador registra los tiempos directamente en la hoja de registro, teniendo alrededor del centenar de horas de desarrollo en total para Abril.

12.1.2 Análisis de los resultados

Los tiempos de *lead* y *cycle* llevan a pensar que el tablero no se está usando todo lo bien que podría. Hablando con el desarrollador, me comenta que le cliente ha estado moviendo a su libre albedrío las

tareas hasta que se ha definido con él un protocolo para que sólo cambie de determinados estados las tareas.

También, otro detalle es la larga vida de las tareas en el tablero. El cliente está definiendo tareas con listas de comprobación que “viven” en los estados de desarrollo del tablero durante demasiado tiempo. Eso “vicia” los datos y ocasione estos valores tan desajustados.

Este proyecto es de desarrollo puro y vemos como es el cliente el que más veces es el que ocasione vueltas atrás de las tareas. En este proyecto es complicado reducir esa cifra ya que las aplicaciones han de ser revisadas por los terceros y usan una serie de criterios que no conocemos a priori. Esto es, el cliente actúa como intermediario.

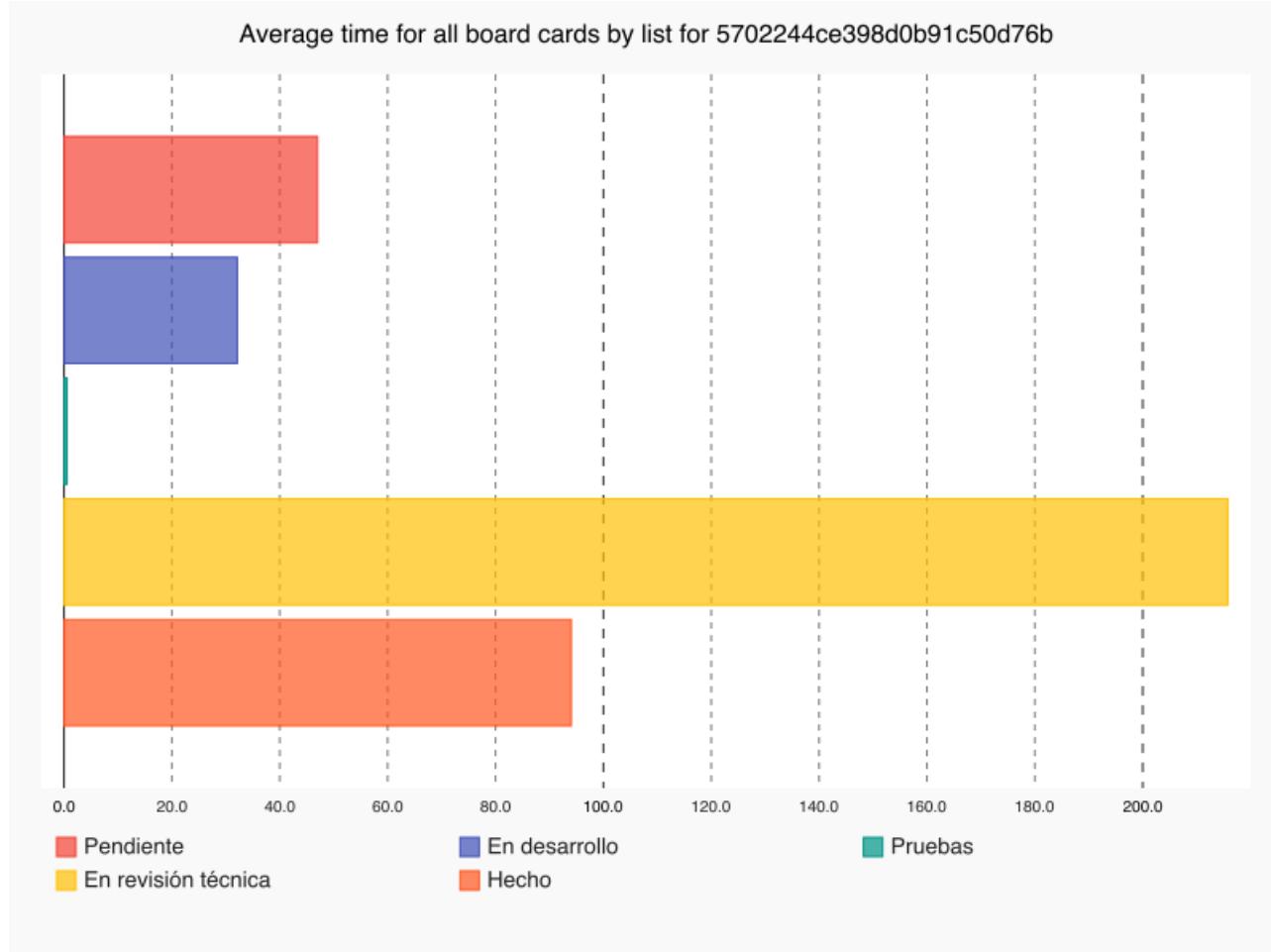
El tiempo de desarrollo equivale a casi tres semanas del mes dedicadas completamente a desarrollo. Esto casa con el hecho de que hay un único desarrollador en este proyecto y está prácticamente dedicado a este proyecto.

12.2 Proyecto B

Este proyecto no tiene limitaciones de otros equipo al ser un proyecto de desarrollo puro, así que los tiempos *lead* y *cycle* no se ven adulterados por los retrasos de otros equipos.

12.2.1 Mediciones

12.2.1.1 *Tiempo medio de las tareas en cada estado*



12.2.1.2 Lead

Media: 277.23 horas.

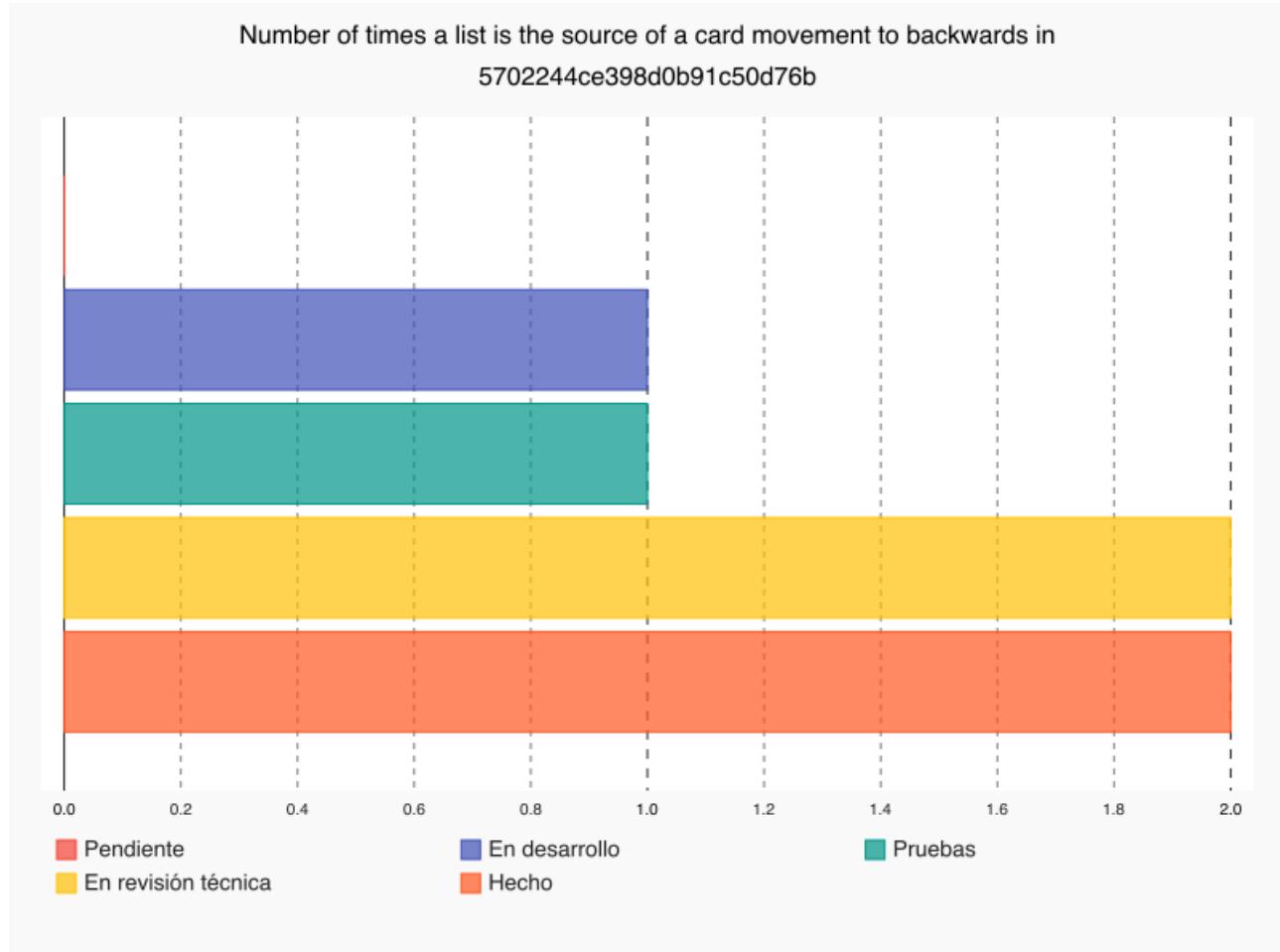
Desviación estándar: 375.22 horas.

12.2.1.3 Cycle

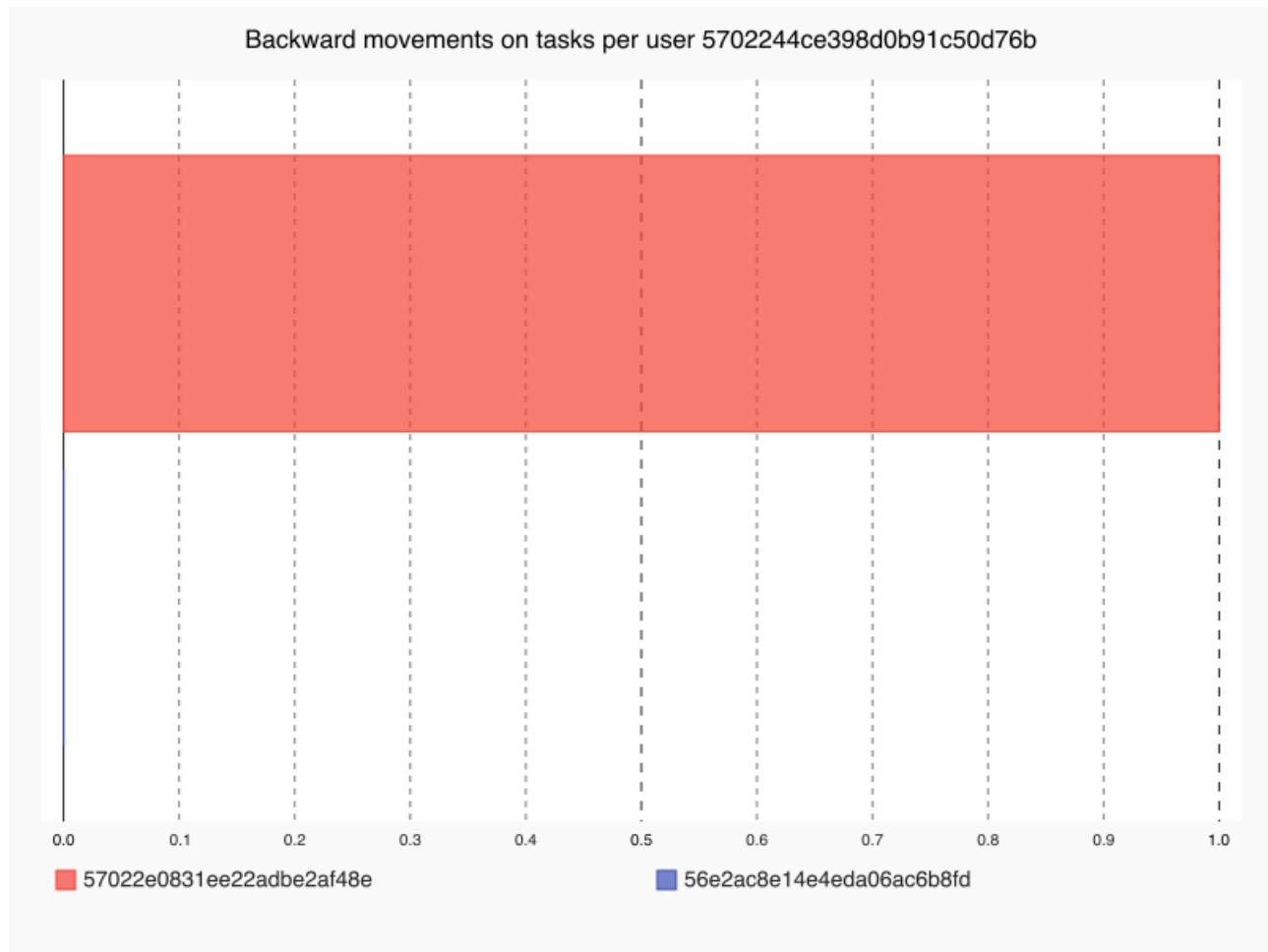
Media: 276.81 horas.

Desviación estándar: 375.1 horas.

12.2.1.4 Estados que ocasionan vueltas atrás



12.2.1.5 Usuarios que generan vueltas atrás



12.2.1.6 Tiempos de desarrollo

Los desarrolladores no han introducido los tiempos de desarrollo completos, de manera que salen valores muy bajos. Revisando los tiempos reales, nos encontramos con que los desarrolladores han consumido 27 horas en Abril y 6 hasta el día 4 de Mayo.

12.2.2 Análisis de los resultados

En primer lugar, vemos la consecuencia de que el equipo tenga muchos proyectos: altos tiempos *lead* y *cycle*. Además, altos tiempos de desviación en estas medidas, ¿por qué ocurre esto? Por la existencia de “ráfagas de trabajo” las tareas se crean y se trabaja sobre ellas, logrando tiempos de vida en el tablero de un par de horas y otros de más de 800 horas. Quizás tendríamos que considerar sólo las tareas sobre las que estamos trabajando y no usar el primer estado como *backlog* o sólo contar el tiempo en el que estamos trabajando sobre el proyecto.

En este proyecto los desarrolladores no han usado correctamente “Plus for Trello” para indicar los tiempos de desarrollo por lo que hemos tenido que revisar la hoja de cálculo de registro del trabajo

por el proyecto. Es imposible interpretar los datos si los desarrolladores no actualizan la información.

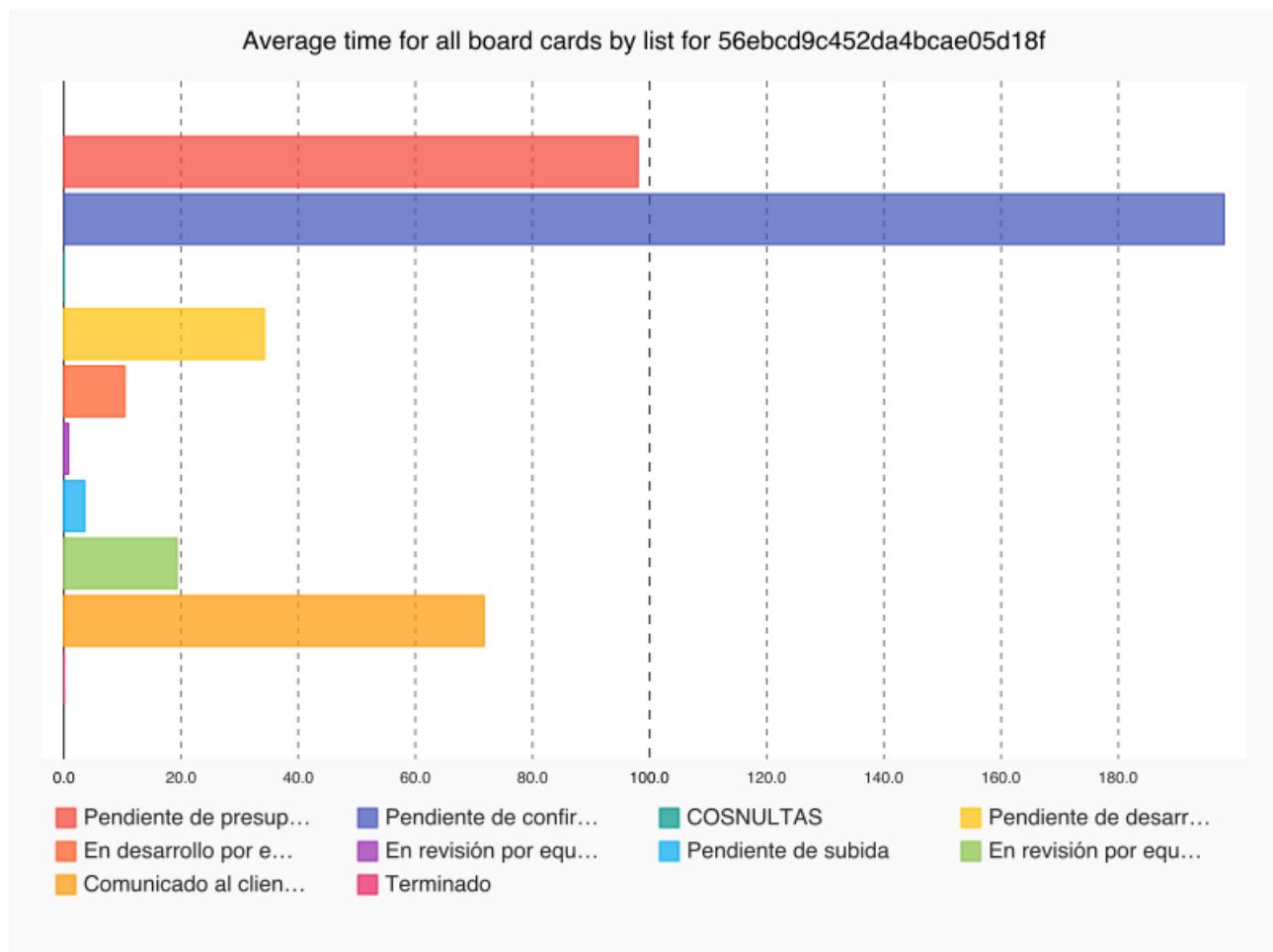
Como sólo hay dos desarrolladores por proyecto y el desarrollador 57022X es el que más trabaja, es normal que él sea el que tenga más pasos atrás.

12.3 Proyecto C

Este proyecto es un sistema de gestión de eventos basado en PHP. Es un proyecto de mantenimiento, por lo que no hay desarrollo constante.

12.3.1 Mediciones

12.3.1.1 Tiempo medio de las tareas en cada estado



12.3.1.2 Lead

Media: 206.63 horas.

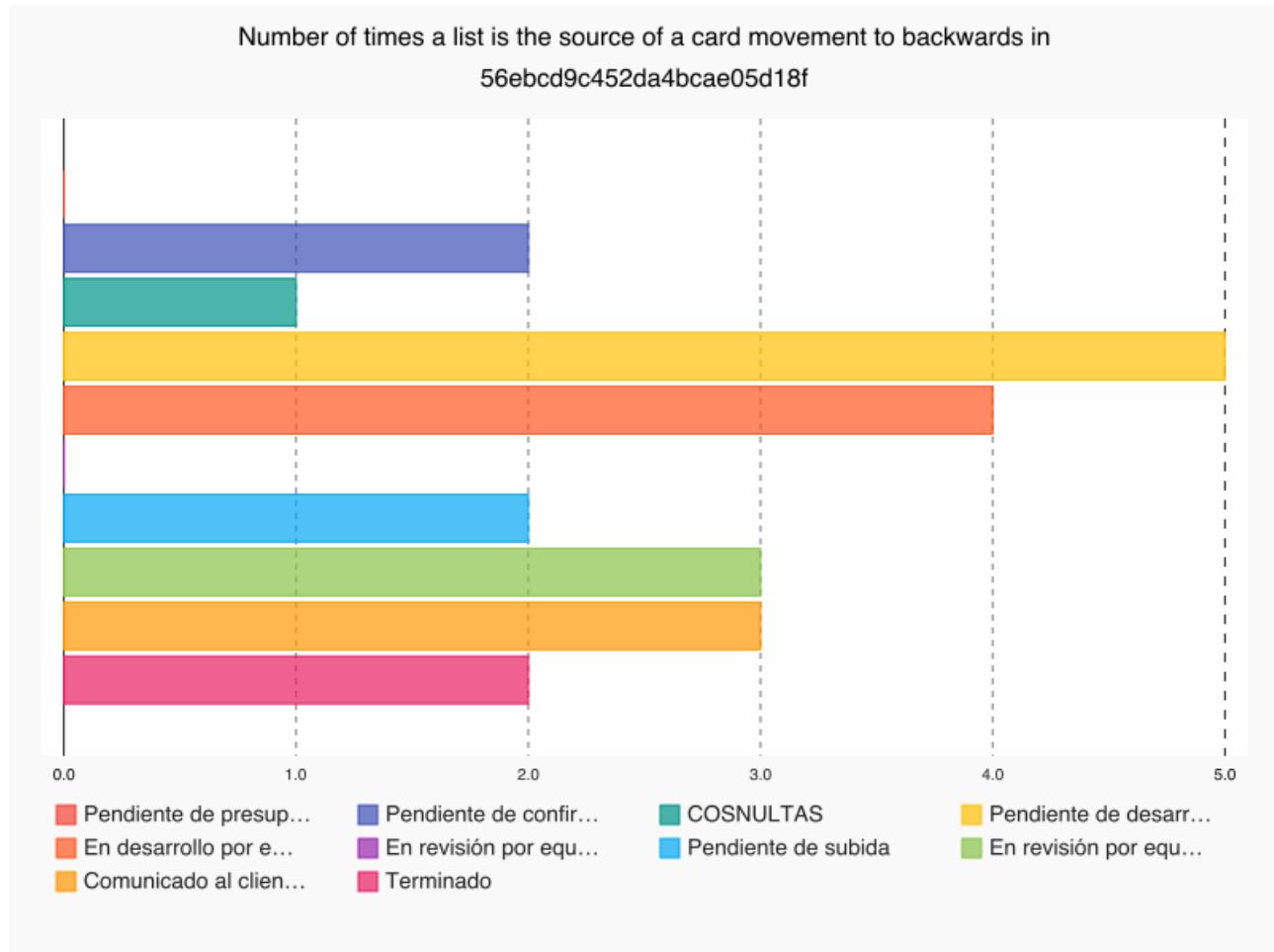
Desviación estándar: 232.87 horas.

12.3.1.3 Cycle

Media: 130.93 horas

Desviación estándar: 171.83 horas.

12.3.1.4 Estados que ocasionan vueltas atrás

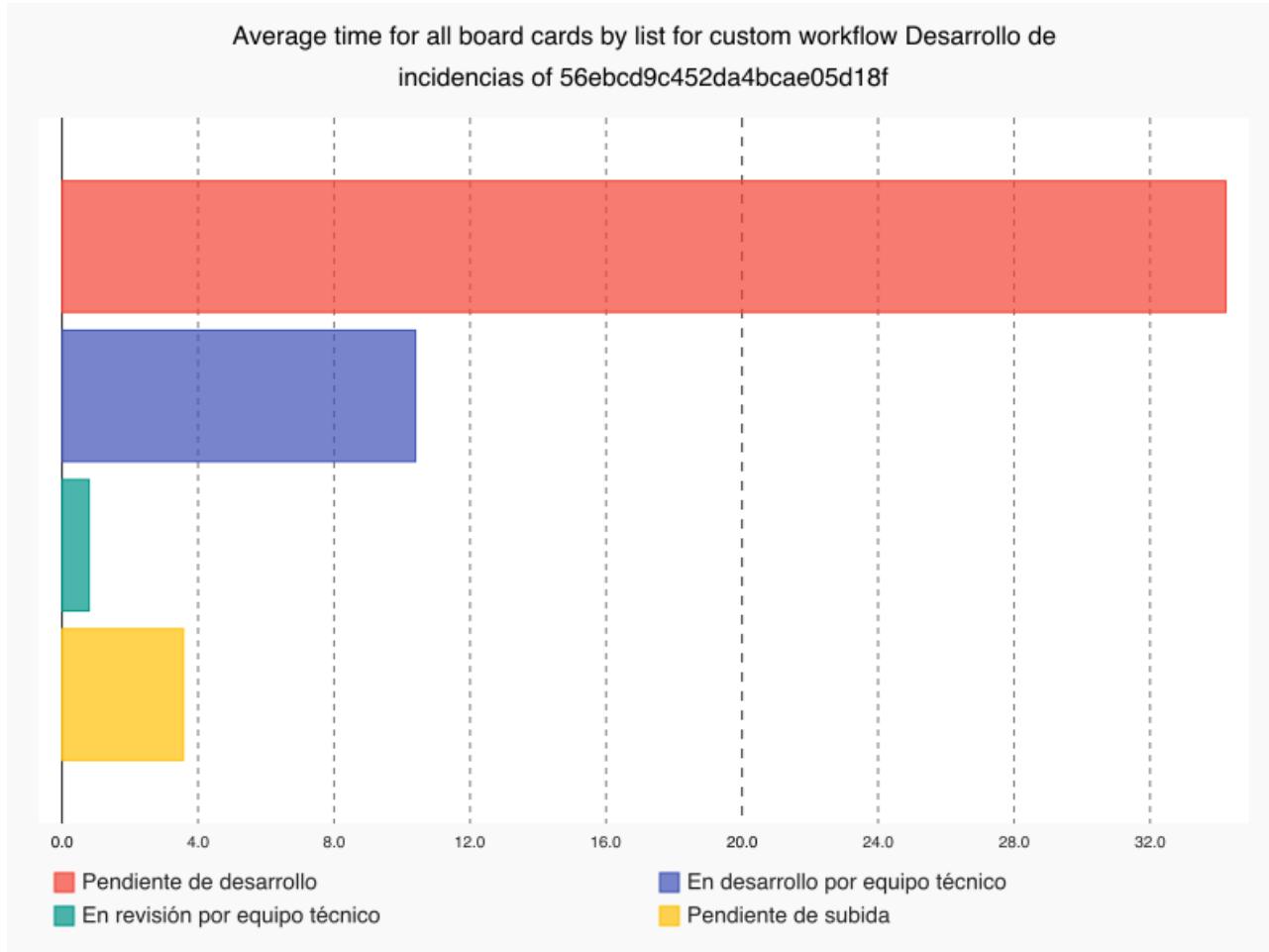


12.3.1.5 Tiempos de desarrollo

En este proyecto los tiempos de desarrollo tampoco se han estado registrando con el complemento Plus for Trello y equivalen en el último mes a 55 horas.

12.3.2 Análisis de los resultados

Los tiempos de *lead* y *cycle* están totalmente desequilibrados por los tiempos de espera en los estados que dependen del equipo comercial. Por esto, hemos definido un gráfico sólo con los estados dependientes del equipo de desarrollo:



Los tiempos de vida de las tareas en este flujo de trabajo son algo más razonables aunque sigue habiendo mucha variación ya que tenemos una media de 63.22 horas (1 semana y media) y una desviación estándar de más del doble de dicho valor (127.52 horas).

Esta gran variabilidad de tiempos de vida las tareas se debe a una serie de tareas de gran tamaño, en especial, una tarea de maquetación de una parte de una web que ha ido avanzando y retrocediendo a lo largo de un mes. En definitiva, tenemos que hacer las tareas más pequeñas para este proyecto.

En el gráfico de los estados que ocasionan una vuelta atrás encontramos que los estados del equipo comercial son los que hacen que las tareas no se consideren completas o que requieran de información del cliente externo y por tanto se muevan atrás.

Es interesante ver cómo sin haber definido un flujo de trabajo al completo, estos casi tres meses de trabajo y la herramienta hacen que se muestre de forma explícita.

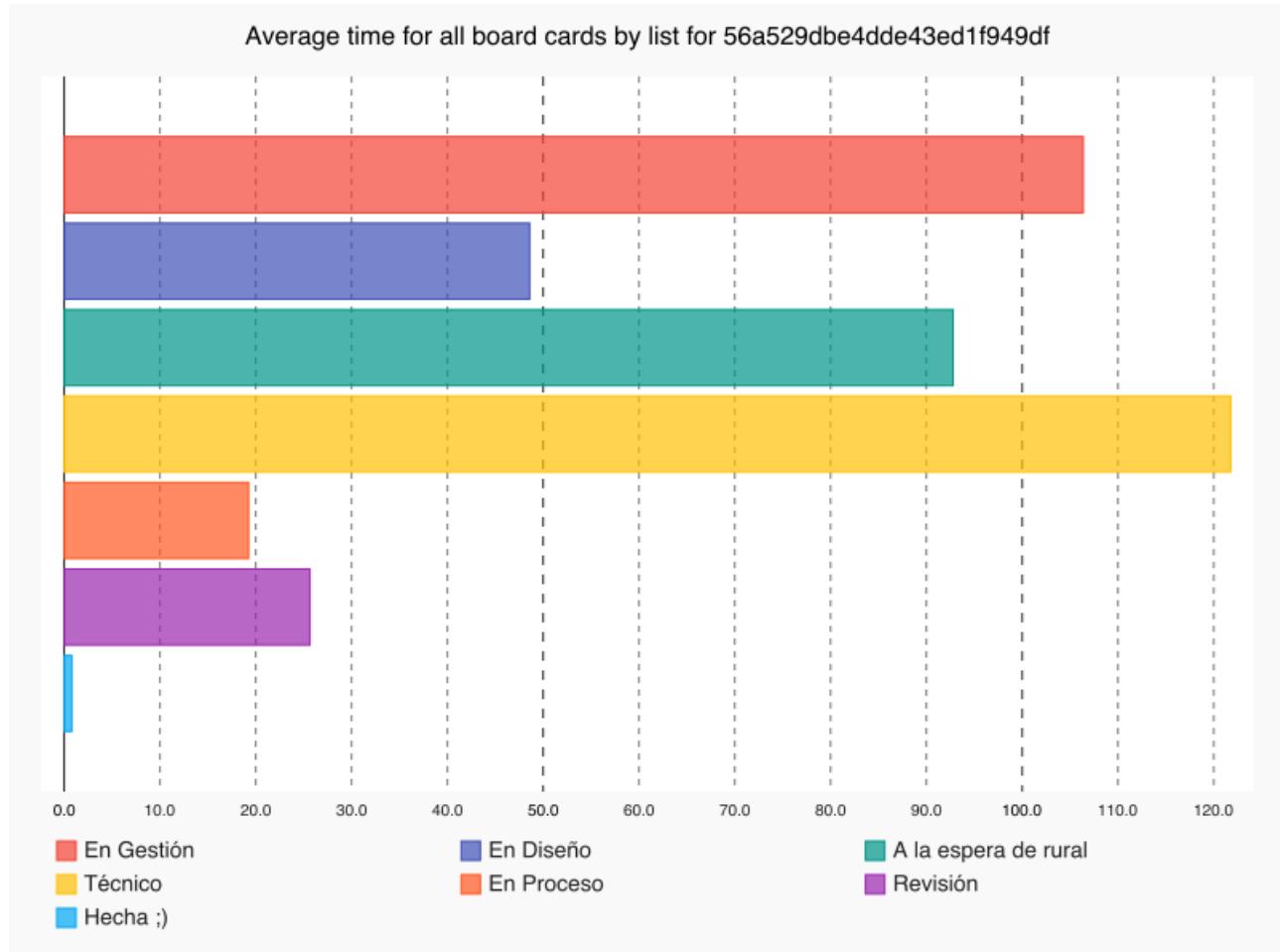
Por último, se va claramente cómo los valores temporales indican labores puntuales de mantenimiento y corrección de errores.

12.4 Proyecto D

Recordemos que este proyecto no es de desarrollo y se basa en gestionar contenidos en una instalación OpenCMS para un tercero. Este proyecto no tiene muchas políticas claras¹⁵ y creemos que los resultados hablarán por si solos.

12.4.1 Mediciones

12.4.1.1 *Tiempo medio de las tareas en cada estado*



12.4.1.2 Lead

Media: 288.52 horas.

Desviación estándar: 461.72 horas.

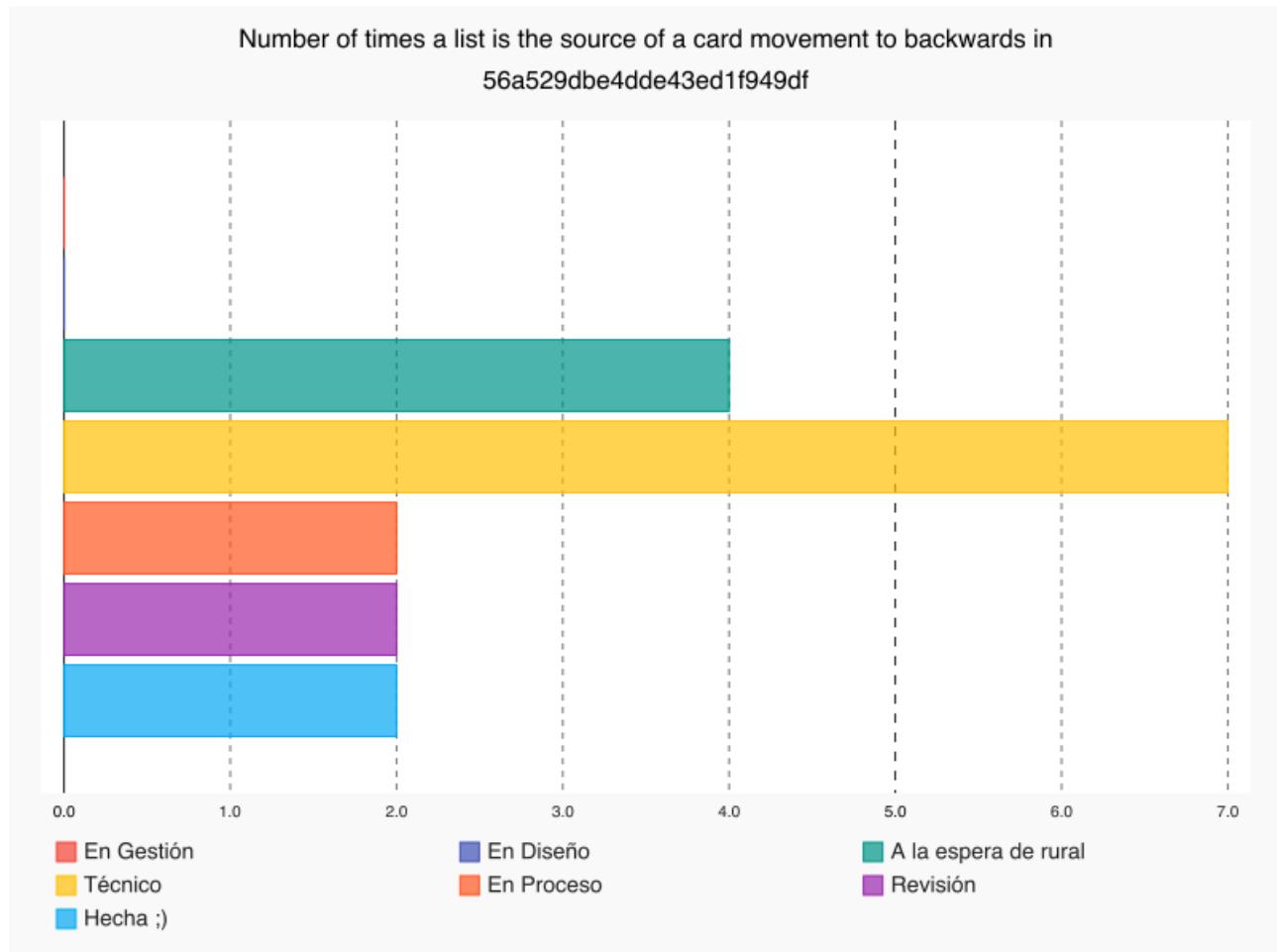
12.4.1.3 Cycle

Media: 62.93 horas.

¹⁵ Técnicamente este proyecto es de otro equipo de desarrollo y nosotros actuamos como soporte y eso se nota en los protocolos y la configuración del tablero Kanban.

Desviación estándar: 181.04 horas.

12.4.1.4 Estados que ocasionan vueltas atrás



12.4.1.5 Tiempos de desarrollo

No tengo acceso a los datos de todos los miembros del proyecto por lo que no voy a poner analizar estos datos.

12.4.2 Análisis de los resultados

En este proyecto se ve cómo no hay una política clara sobre las tareas. De hecho, las tareas tienen tamaños realmente desequilibrados, lo que se nota en los tiempos *lead* y *cycle*.

Me ha llamado la atención los grandes tiempos de espera que tienen las tareas en los estados anteriores al de desarrollo de las acciones técnicas. Sé que el equipo está teniendo problemas con los clientes para establecer un protocolo y las revisiones adecuadas de las acciones por lo que entiendo que estas son las consecuencias.

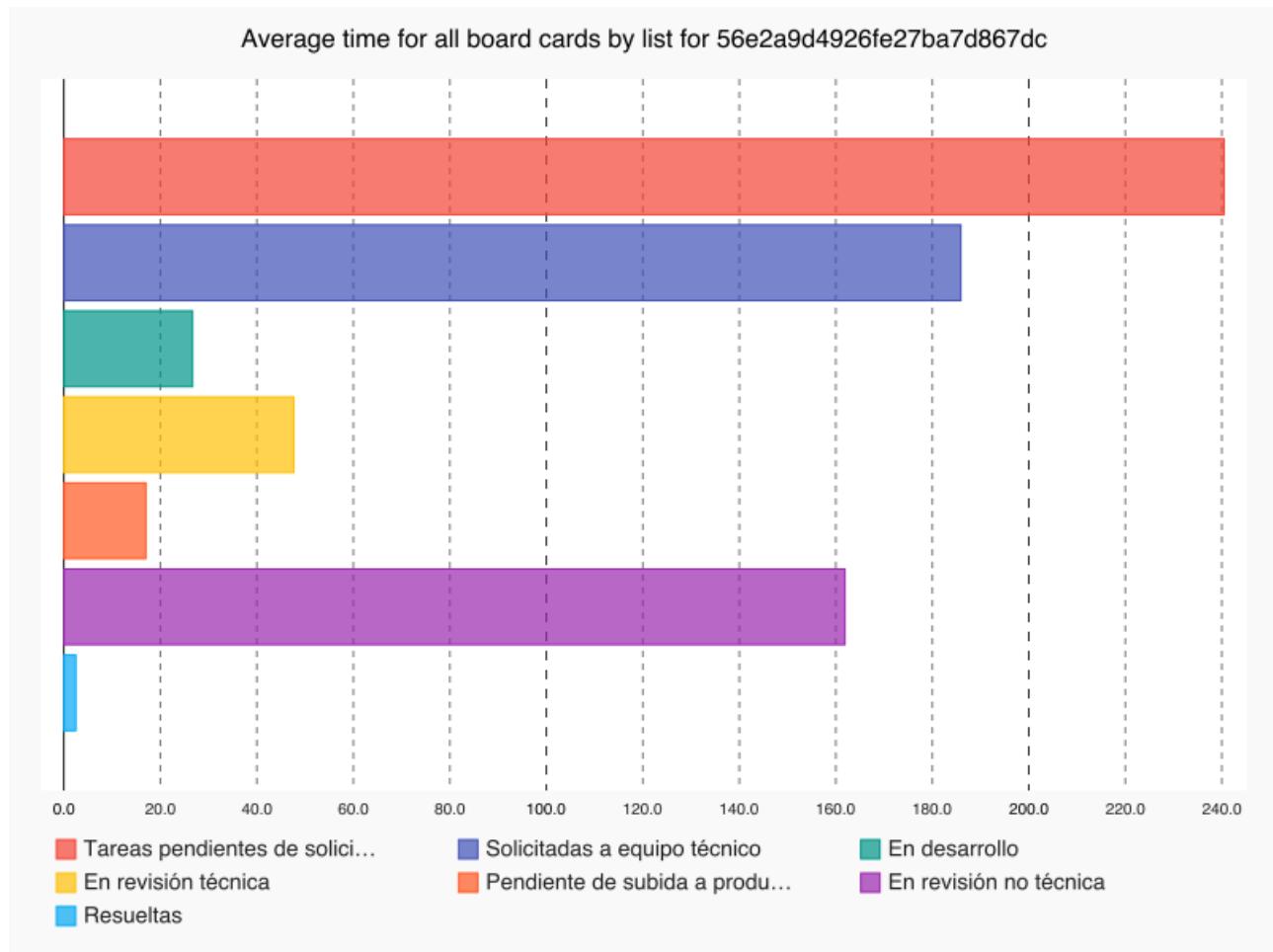
12.5 Proyecto E

Recordemos que este proyecto es el de mejora de la web de la empresa. Y casi todas las acciones son de maquetación de HTML con CSS, por lo que se podrán definir y estimar de forma sencilla.

También, al ser un proyecto interno, no hay tareas inesperadas y éstas están muy bien definidas.

12.5.1 Mediciones

12.5.1.1 *Tiempo medio de las tareas en cada estado*



12.5.1.2 Lead

Media: 605.03 horas.

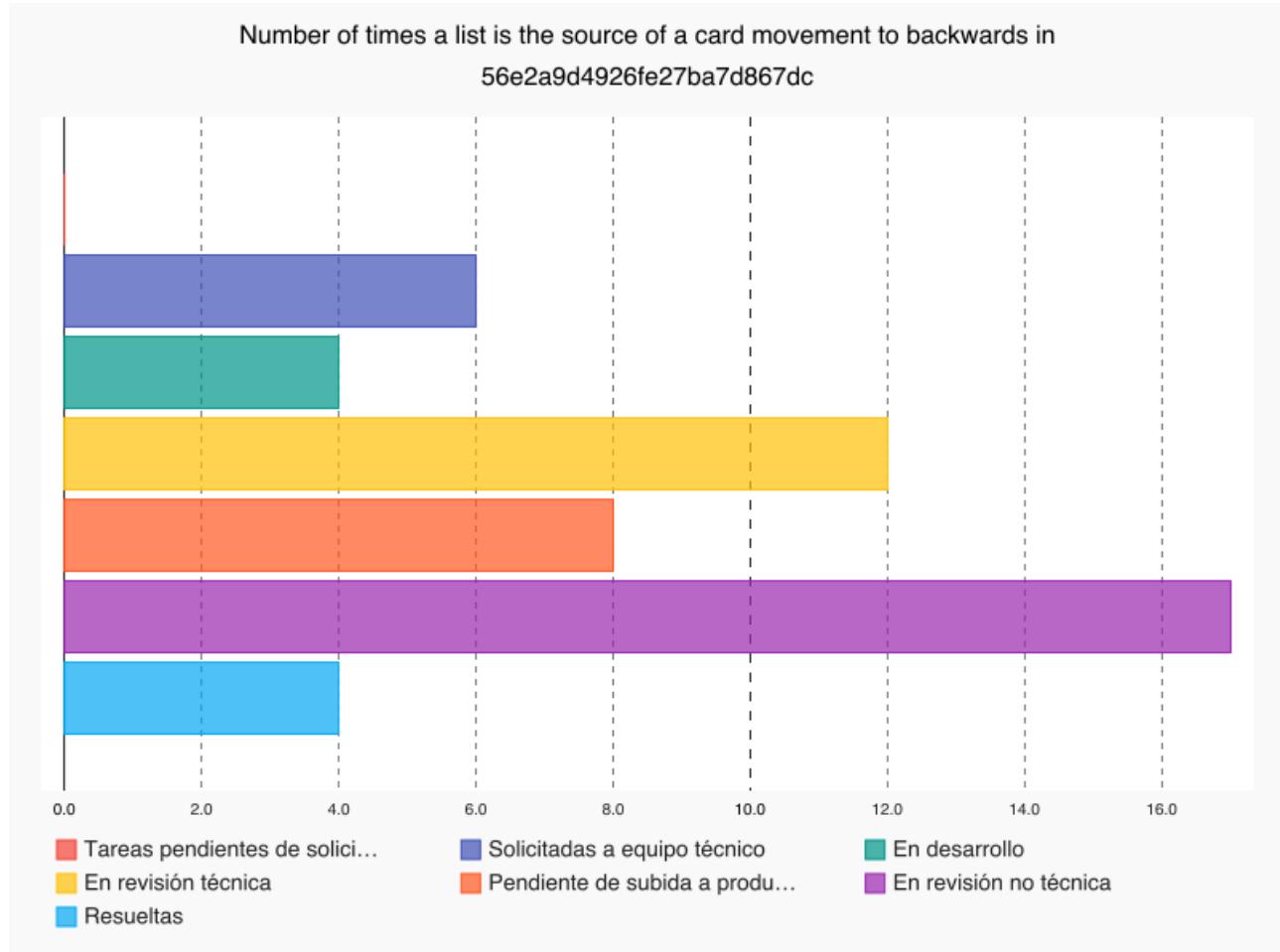
Desviación estándar: 344.97 horas.

12.5.1.3 Cycle

Media: 340.65 horas

Desviación estándar: 245.42 horas.

12.5.1.4 Estados que ocasionan vueltas atrás



12.5.1.5 Tiempos de desarrollo

Los tiempos de desarrollo para el mes de Abril son unas 50 horas. En el tablero sólo se han registrado 11 horas. Al igual que ocurre en los otros proyectos, los miembros del equipo no están usando la herramienta “Plus for Trello”.

12.5.2 Análisis de los resultados

Lo primero que nos llama la atención es ver cómo el cuello de botella de las tareas es el departamento de comunicación, ya que éste es el encargado de la revisión no técnica. También vemos cómo los dos primeros estados “Tareas pendientes de solicitar” y “Solicitadas a equipo técnico” tienen tareas con mucho tiempo de vida ahí. Esos estados son el *backlog*.

En este proyecto se nota la buena colaboración del departamento de comunicación a la hora de definir tareas de un tamaño adecuado. También la naturaleza de éstas las hace más similares en tiempo y eso lo vemos en los tiempos *lead* y *cycle*. Estos tiempos son realmente grandes debido a que el departamento comercial creó un gran *backlog* de tareas.

Eliminando los estados dependientes de otros departamentos, obtenemos un tiempo *lead* medio de 539.06 horas con una desviación típica de 300.18 horas porque hay tareas de 1000 horas. Todavía tenemos que investigar cómo puede ser que haya tareas que consuman tanto tiempo mientras que la mayoría tiene un coste más bajo y sobre todo, más similar.

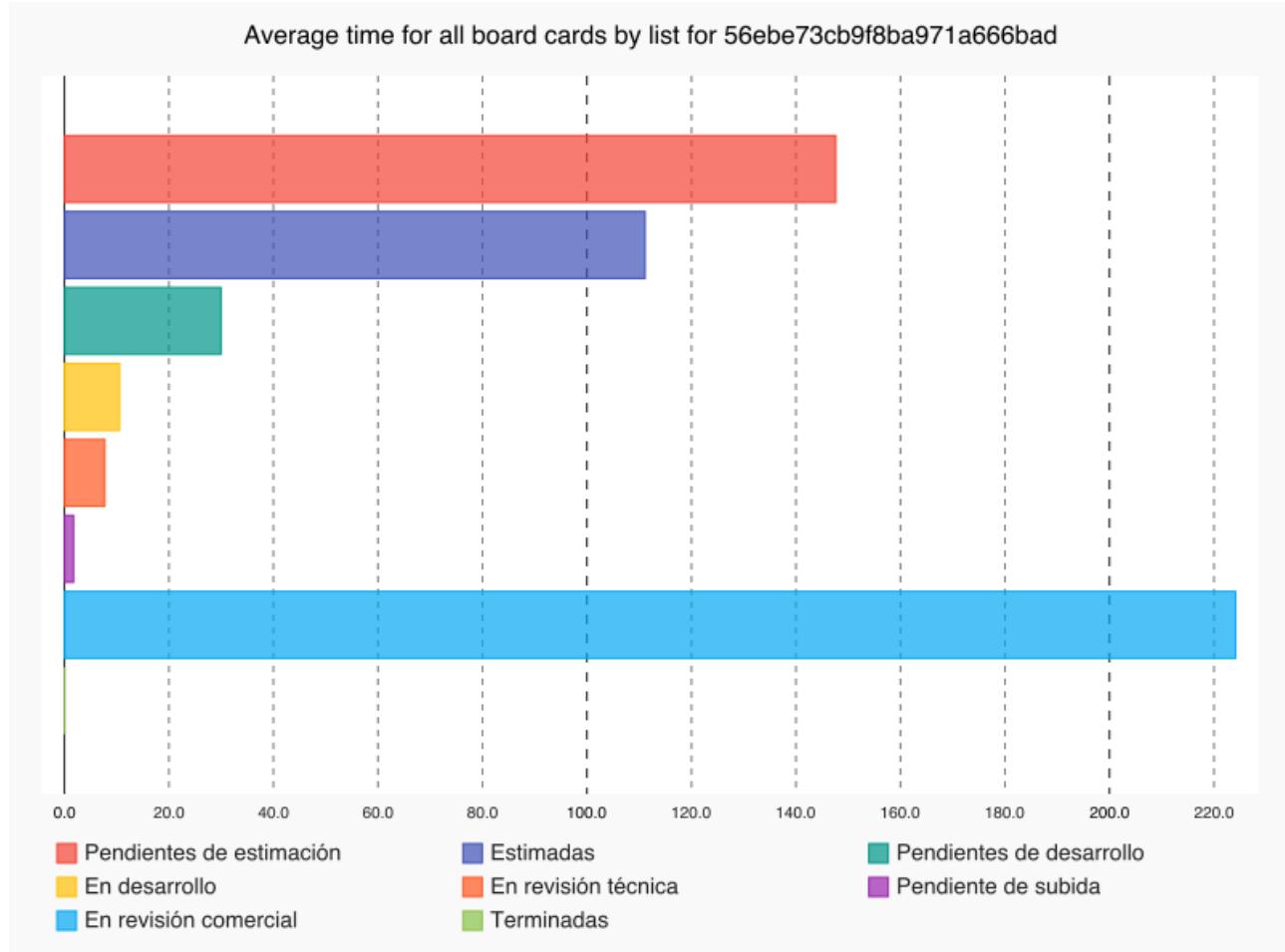
La revisión no técnica es el estado que más veces hace que las tareas tengan que volver, seguido por la revisión técnica. Esto es normal en el sentido de que los estados de revisión son los que deben mover atrás las tareas que no estén completamente terminadas o que no sean correctas.

12.6 Proyecto F

Esta es la aplicación de venta de entradas. Ya está desarrollada y sólo requiere nuevas implantaciones y pequeñas labores de mantenimiento.

12.6.1 Mediciones

12.6.1.1 Tiempo medio de las tareas en cada estado



12.6.1.2 Lead

Media: 318.83 horas.

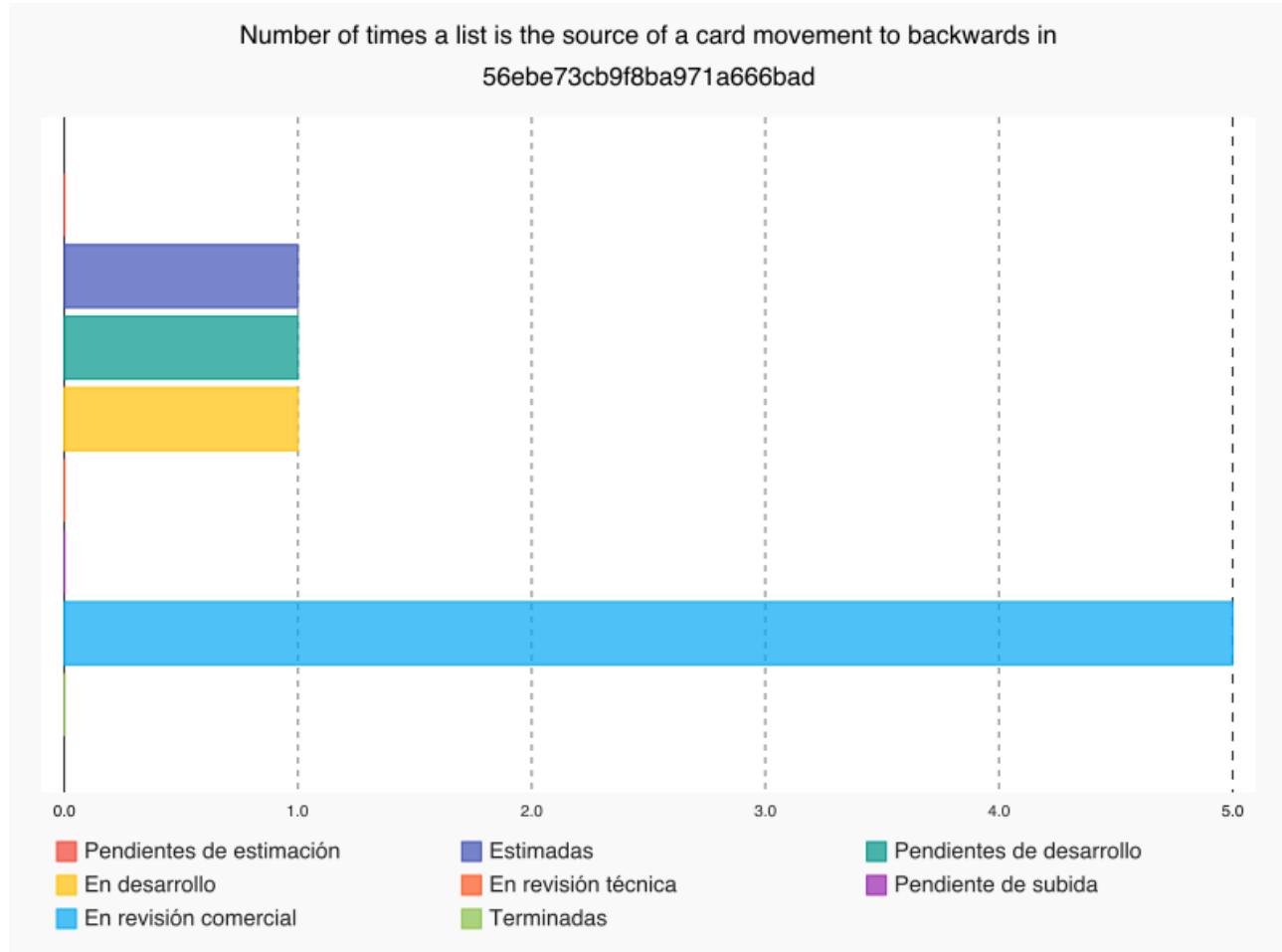
Desviación estándar: 272.12 horas.

12.6.1.3 Cycle

Media: 142.98 horas.

Desviación estándar: 206.78 horas.

12.6.1.4 Estados que ocasionan vueltas atrás



12.6.1.5 Tiempos de desarrollo

Al igual que en los otros proyectos, no se está usando el registro de tiempos estimados y consumidos en el desarrollo de cada tarea. El último registro indica que en Abril se usaron 20 horas.

12.6.2 Análisis de los resultados

Este proyecto tiene cuellos de botella en el departamento comercial debido a la complejidad del modelo de negocio. Se puede ver claramente en el hecho de que las tareas pasan más tiempo en los estados de “Pendiente de estimación”, “Estimadas” (y esperando aprobación para comenzar el desarrollo) y en el estado “En revisión comercial”. El equipo comercial no da un flujo constante de

tareas para este proyecto y usa algunos estados como *backlog*.

Los tiempos *lead* y *cycle* sufren el mismo problema que el resto de proyectos ya que hay mucha variabilidad. Si vemos sólo los estados de desarrollo, su tiempo *lead* es de 72.66 de media y con una desviación estándar de 105.91. Esto es, tardamos una media de dos semanas en sacar una nueva funcionalidad para este proyecto.

El estado que más veces hace retroceder a las tareas de estado es el de “En revisión comercial”. Esto se debe a lo poco detalladas que están las tareas por el equipo comercial, de manera que preferimos trabajar con pequeños prototipos e ir desarrollándolos según nos vayan dando retroalimentación.

12.7 Conclusiones

En primer lugar, hemos de decir que no hemos podido hacer un estudio tan exhaustivo como nos hubiera gustado.

No hemos podido medir el número de incidencias debido a que los equipos que nos las comunican no han usado etiquetas con algún texto estándar que indique cuáles tareas son incidencias y cuáles son mejoras. Este punto queda pendiente para el futuro.

Con los tiempos *lead* y *cycle* tan altos que tenemos, no tenía sentido calcular el número de tareas por hora, ya que se llegaría a valores ínfimos. Esto se debe al gran número de proyectos que tenemos. Esto es, las tareas de los proyectos esperan mucho tiempo a que tengamos un hueco para poder desarrollarlas. De igual forma, por eso se obtienen desviaciones tan altas en todas las mediciones.

En especial, hemos detectado que el equipo comercial hace uso de la herramienta indicando varios estados como *backlog*. Tendremos que tener en cuenta esto para el futuro y definir flujos de trabajo que ignoren dichos estados.

También, hemos notado como el tiempo de las revisiones comerciales es bastante elevado y cómo el estado de “En revisión comercial” suele ser el que más devuelve tareas a otros estados anteriores. Parece que está habiendo un fallo en la captura de requisitos por su parte.

Otro aspecto que hemos echado en falta es mayor volumen de datos. Estamos seguros que con un mayor intervalo de tiempo los datos serían más representativos. Esperamos poder repetir las mediciones en 6 meses y ver si las métricas han mejorado.

Por último, la velocidad de desarrollo no se ha podido medir tampoco debido a que el equipo no ha hecho un buen uso de la herramienta Plus for Trello, por lo que los datos han sido incompletos. Se centrará el foco en el uso de esta herramienta en un futuro.

13 Evaluación personal del proceso de mejora

El principio del proceso fue duro por el desconocimiento sobre los proyectos que se me habían asignado y sobre el equipo. Los proyectos eran desconocidos (y sin ninguna documentación) y sobre el equipo no tenía mucha información sobre sus fortalezas y debilidades.

También, el hecho de tener varios proyectos retrasados hacia que el estrés estuviera en niveles más elevados de lo que me hubiera gustado.

Por otro lado, al principio me sentía que tenía que hacer el doble de trabajo: gestionar los proyectos y desarrollar. Pero mi intención era que hubiera un proceso de guía al principio que luego me permitiese que mi equipo actuase de forma autónoma. Esto es, guiarlo al principio, formándolo en cómo desarrollar software de calidad y cómo seguir el marco de trabajo Kanban. Una vez que tuvieran cierto manejo, yo podría dejar de estar presente en las decisiones de peso y dejar que ellos se autoorganizasen.

El proceso de formación era muy necesario debido a la existencia de malas prácticas de desarrollo. Muchos de los desarrolladores hacían código ilegible, ineficiente y poco mantenible. El primer paso era vencer esa tendencia y para ello tuve que dedicar muchas horas a revisiones de código y a desarrollo por parejas. Poco a poco el nivel de desarrollo ha ido creciendo hasta crear código entendible, mejor estructurado y, en definitiva, de mucha mejor calidad.

Este proceso supuso un desgaste personal muy fuerte. El director ejecutivo incluso llegó a hablar conmigo sobre este detalle, casi temiendo por mi salud, dadas las largas jornadas de trabajo a las que yo me sometía. Porque además de ser “el soporte” de mi equipo, yo también tenía proyectos asignados y sólo podía dedicarme a ellos cuando los miembros de mi equipo se habían marchado, por lo que trabajaba demasiadas horas.

Para colmo, cuando llegaba a casa después del trabajo, desarrollaba la herramienta software PyStats-Trello [PYSTTRLO] y estudiaba los resultados, buscando qué métricas eran las mejores para extraer información de rendimiento sobre el equipo de trabajo en cada uno de los proyectos.

Esto, ha ido cambiando conforme el equipo se ha ido formando y ahora el equipo es prácticamente autónomo. Ya todos colaboran en los proyectos y saben resolver los problemas por sí solos. Usan Kanban sin problemas e incluso se comunican con los dueños de producto sin necesitar de mucho apoyo por mi parte.

Es destacable también el hecho de querer formar parte de la comunidad de desarrolladores. En ese sentido, asistimos a una charla sobre *Test Driven Development* y esperamos asistir a más reuniones y charlas de este tipo.

Creo que todos somos patentes del gran cambio que hemos dado. Todo el equipo ha crecido como trabajadores y desarrolladores (en particular) y estoy muy orgulloso de ellos.

El camino está marcado. Creo que si seguimos en esta línea, podemos aportar mucho a la empresa y

tener un desempeño en nuestro trabajo que nos haga capaces de abordar proyectos mucho más complejos y por supuesto, más interesantes.

En definitiva, todo mi trabajo, las largas jornadas y mi esfuerzo se ha visto recompensada con creces tanto a nivel de rendimiento del equipo como a nivel personal. He aprendido mucho sobre gestión de clientes, gestión de recursos humanos y gestión del cambio. No cambiaría esta experiencia por nada del mundo.

Ahora, se presenta un horizonte fascinante en el que espero que los desarrolladores de mi equipo se formen en distintas tecnologías y podamos aportar mucho más valor a la empresa. Porque veo en ellos interés, ganas de mejorar y mucha motivación para continuar con este proceso. Eso es lo importante. A fin de cuentas, eso es el *kaizen*.

14 Conclusiones finales

14.1 Sobre la aplicación del proceso de mejora Kanban

14.1.1 Los cambios en organizaciones tardan tiempo

Lo primero que he visto es que los cambios tardan tiempo en tener efecto en una organización. Esta idea no es nueva. David J. Anderson ya la indica en una de los análisis de mejora de Kanban en [ANDERS10].

A lo largo de algo más de 3 meses que se me ha dado la oportunidad de gestionar un equipo, he conseguido hacer bastantes cambios en la cultura, flujo y calidad del trabajo. Pero reconozco que una evaluación a largo plazo, en un año, es totalmente necesaria para ver la evolución de este proceso.

Para ello, tendré que repetir los cuestionarios de análisis de la profundidad de Kanban, los de satisfacción y el estudio de métricas por cada uno de los proyectos en el año 2017. Seguramente los proyectos sean distintos a los actuales, pero por lo demás, será una gran forma de evaluar este proceso.

14.1.2 Siempre hay gente que se resiste a un cambio

Los seres humanos somos reacios a los cambios. Nos gusta la rutina, la seguridad de lo conocido. Si no lo vemos completamente necesario, no cambiamos.

Anderson dice en [ANDERS10] que un *pleado sólo aceptará los cambios cuando vea de forma clara que le reportan algún beneficio*.

En el caso de este proceso de mejora, he comprobado de buena manera que es así. Varios miembros del equipo comercial se sintieron molestos por la nueva política de evitar las interrupciones. En ningún momento quería yo eliminar la comunicación entre desarrolladores y comerciales, pero considero (y creo que con razón) que hay mejores formas de hacerla más allá de acercarse al puesto e interrumpir el trabajo del desarrollador.

Hasta que no he recibido apoyo explícito de la dirección, los individuos que generaban las interrupciones, ignoraban completamente mis recomendaciones sobre este asunto, e incluso se sentían “atacados” por mí según me hicieron llegar en varios correos electrónicos.

Para ellos, este cambio no mejoraba su forma de trabajar, porque les obligaba a ser ordenados y enviar la información por correo electrónico. Da igual que les justificase (incluso con trabajos científicos como [PARMIN2012]) el mal que hacen las interrupciones a los programadores. Ellos no ven beneficio en cambiar su trabajo, el que lo haya para otros no les importa.

14.1.3 La calidad software ha de ser lo primero

Dice David J. Anderson en [ANDERS10] que el primer paso en aplicar Kanban es hacer software con calidad. Por eso este proceso de mejora que he empezado, comenzó precisamente con un

proceso de formación basado en cursos gratuitos en línea y libros clásicos como *Clean Code*, [MARTIN08].

De hecho, uno de nuestros proyectos tiene una alta deuda técnica y vemos reflejado esta enseñanza en él todos los días.

14.1.4 La formación del equipo es fundamental

Independientemente de que se sea desarrollador o ingeniero, en nuestra profesión hay que estar cada día formándose. No sólo en tecnologías, sino en marcos de trabajo y otros conocimientos y habilidades relacionadas. El desarrollador que no se forma, se queda, irremediablemente, atrás.

Por eso he hecho tanto énfasis en que mi equipo siguiera cursos en línea y estudiase material que yo les iba proporcionando. Todos los miembros debemos tener un nivel similar para poder entendernos entre nosotros y gracias a este proceso, las distancias entre los distintos desarrolladores se han reducido.

En ese sentido, mi equipo no me ha defraudado y todos han mejorado mucho en este proceso.

14.1.5 La motivación es parte importante del proceso

Programar es una actividad intelectual compleja, [DEMARCO99] y si no se tiene una motivación y moral para “luchar” contra los problemas a los que nos enfrentamos día a día, no se trabajará de forma eficiente.

Evidentemente, la motivación y el bienestar del programador es importante. El estado anímico es muy importante en casi cualquier actividad, pues más aún en una que usa tanto el recurso más valioso que tenemos: el cerebro.

14.1.6 El objetivo es la mejora constante: el *kaizen*

Kanban proporciona a nuestro equipo transparencia. Esto no es nada nuevo, en [ANDERS10] se hace mención a este hecho. Además, teniendo en cuenta la confianza que tanto la dirección como yo mostramos en su trabajo y en sus capacidades para autoorganizarse, hacen que sea más sencillo que el equipo alcance el *kaizen*, esa cultura de mejora constante que puede ser la diferencia entre desarrollar un proyecto o hacer que un proyecto tenga éxito.

La diferencia básica es que los desarrolladores no sólo hacen lo que se les dice, y en dos sentidos aportan retroalimentación al proyecto.

Primero, tratan de comprender los flujos de trabajo del negocio cuyo software estamos desarrollando y si habría otro proceso de negocio mejor. Por lo tanto, aportan retroalimentación al cliente.

Segundo, tratan de encontrar las mejoras herramientas disponibles para los requisitos del proyecto y eso origina no sólo retroalimentación, sino una vía de comunicación nueva entre desarrolladores, dirección técnica e incluso cliente.

En definitiva, es la diferencia entre aportar valor al cliente o no aportar más que tu trabajo.

14.1.7 Kanban no es sólo usar un “tablero de tareas”

Aplicar Kanban consisten en cambiar la mentalidad de un enfoque secuencial (en cascada) a uno basado en la adaptación a los cambios y en la Teoría de las Restricciones [BARNARD]. Esta teoría busca identificar las restricciones del sistema y centrarse en eliminarlas poco a poco. Es un enfoque científico a la gestión que es el que aplicar David J. Anderson en [ANDERS10].

De hecho, si aplicar Kanban hubiera sido sólo usar un tablero de tareas, varios de los equipos de la empresa en la que trabajo ya lo habrían estado haciendo. Elementos como el límite del WIP, la mejora continua, la transparencia y visualización total, la formación e incremento de la calidad de forma continua, el estudio de los cuellos de botella y las métricas de Kanban son los que marcan la diferencia entre el método y el marco completo de trabajo.

14.1.8 Rendimiento del trabajo

Los proyectos tienen grandes tiempos de espera para las tareas. Tenemos multitarea de proyectos y eso no bueno. ¿Qué ocurre cuando llegan dos incidencias críticas a la vez?

Debemos definir políticas más claras y un límite WIP no por estado y proyecto, sino general. No podemos tener tantas tareas abiertas como queramos e ir cambiando de proyecto. Hemos tratado de aplicar un límite de WIP por estado proyecto pero en realidad, parece que el problema no está ahí sino en la multitarea a nivel de proyecto.

Tendremos que plantearnos usar límites WIP adicionales por persona, de manera que una persona no pueda pude trabajar sobre más tareas de un límite en un momento dado. Sean las tareas del proyecto que sean.

14.1.9 Métricas de Kanban

Necesitamos tener métricas objetivas que nos indiquen el rendimiento del equipo de forma cuantitativa.

Ahora mismo esto no es posible, ya que como tenemos muchos proyectos, los tiempos de vida de las tareas en los tableros no son relevantes más allá de saber pronosticar cuándo vamos a poder sacar una característica al mercado. Pero no nos sirven para saber el tiempo consumido en media. Esto es, el tiempo de vida de la tarea no se corresponde con el tiempo consumido por el desarrollador en completarla.

Para ello tenemos que comenzar a usar alguna herramienta de medición de tiempo efectivo en cada tarea de una forma más adecuada. Hay que usar “Plus for Trello” de forma más intensiva para registrar los tiempos de estimación de las tareas, y sobre todo los de desarrollo de éstas.

14.1.10 Midiendo el desperdicio

Una de las ideas más interesantes que he encontrado en [ANDERS10] es la de medir los costes de coordinación y de transición del equipo. Los costes de coordinación son aquellas tareas que proporcionan sincronización y la base de comunicación del equipo. Los costes de transición son aquellos que se generan al ir pasando las tareas por los distintos estados hasta llegar a la entrega, o

lo que nosotros llamamos “producción”.

Los costes de producción son fundamentalmente las reuniones diarias y las de retrospectiva, y dado que somos muy regulares en la realización de éstas, su coste es sencillo de medir.

Los costes de transición son más complejos porque dependemos del equipo de administración de sistemas y no hay planes de entrega ni actualización de los servidores de producción. Así, tenemos que establecer una estrategia para estudiar cómo nos afecta ese pequeño retardo en la entrega a los clientes de las funcionalidades desarrolladas.

Realizaremos estas mediciones en un futuro para poder evaluar de forma más completa el rendimiento del equipo y el nivel de “desperdicio” que estamos generando.

En ese sentido, dos de los proyectos en los que no hemos podido aplicar Kanban y que estaban generando pérdidas se han contabilizado como desperdicio al 100%. Lo bueno es que los hemos terminado y ya interrumpirán el proceso de mejora más.

14.1.11 Políticas con los equipos con los que colaboramos

Los picos de trabajo y no uniformidad de los tamaños de las tareas está haciendo que sea complicado interpretar las mediciones. Tenemos que definir los estados de cada equipo y tener flujos de trabajo distintos, con sus mediciones propias y sus métricas para poder optimizar el proceso.

Los picos de carga de trabajo han de eliminarse y esperamos del equipo comercial que sea capaz de definir una cadencia de tareas (cuando sea posible).

Las tareas han de ser de menor tamaño (por lo general) y esperamos que la experiencia haga que se vayan pareciendo sus tamaños cada vez más.

PyStats-Trello permite definir distintos flujos de trabajo por cada proyecto, por lo que desde el punto de vista técnico no hay problema en usar algunos estados como *backlog*.

14.1.12 Colaboración con clientes

Hasta ahora mismo sólo un cliente está usando la herramienta de Kanban como tablero de visualización de tareas. A otro se la hemos presentado y parece que tiene interés.

Tenemos que hacer que el resto de clientes colaboren, no sólo en el uso del tablero (que sería lo mínimo) sino en el proceso al completo.

En el software trabajamos con ideas, y si no recibimos retroalimentación constante de los clientes, los proyectos no se ajustan a sus necesidades, el proyecto fracasa y el cliente no está contento.

14.2 Personales

14.2.1 Investigación en ingeniería del software real

Siempre he intentado hacer esfuerzos en el sentido de aproximar la investigación en Ingeniería del Software con el desarrollo en el mundo real. Lo intentó anteriormente con [DJBDD] y comenzando

a estudiar Teoría de Categorías [BARR98], programación funcional [OCAML] o *Model-checking* [BAIER07]. Pero no conseguí aportar valor a la empresa de una forma sustancial primero debido a la falta de una guía efectiva y segundo, debido al gran esfuerzo que supone profundizar en este tipo de técnicas y segundo a su limitada aplicación en una PYME.

Es cierto que mis conocimientos técnicos crecieron gracias a esta formación, pero no conseguía lograr una mejora en el ámbito de la ingeniería del software.

Ahora creo que he dado un paso de gigante en conocer métodos de gestión científica para mejorar la productividad y calidad de un equipo o empresa dedicados al desarrollo software.

14.2.2 La gestión de proyecto software es compleja

Desde que entré en la empresa he ido evolucionando y, con la experiencia que tengo, he visto que la mayoría de los problemas que hay en los proyectos son problemas humanos. Bien por falta de motivación, formación u organización de los desarrolladores. O bien por parte de los clientes que no comprenden como funciona el proceso de desarrollo de software.

He aprendido muchas técnicas en este proceso, pero no sólo eso. Esta experiencia me ha abierto los ojos sobre lo que es un gestor y lo que es un líder de equipo. Nunca quise que mi equipo me viera como un gestor o un jefe, sino como alguien que estaba ahí para ayudarlos y que trabajaba como el que más. Creo, según las encuestas que me han contestado que así es, por lo que puedo valorar este proceso como un éxito.

14.2.3 Un reto a superar

En definitiva, este proceso ha supuesto un reto personal, laboral y académico.

El reto personal era hacer un trabajo que reflejase un proceso de mejora en mi lugar de trabajo en vez de hacer una simulación. Me considero una persona práctica y quería hacer algo útil.

Por otro lado, a nivel laboral, había intentado varias veces hacer un proceso de mejora en mi trabajo con escaso éxito. El motivo de esto era mi desconocimiento de los fundamentos de la dirección y gestión de proyectos software. Dado que este máster me ha dado esos conocimientos, veía que era una oportunidad de ponerlos en práctica.

Por último, a nivel académico desde que comencé a trabajar en Ingeniería del Software me he planteado cómo aunar mis conocimientos con mi trabajo. Este máster me ha demostrado que se puede estudiar cómo mejorar los procesos de desarrollo de software de una organización y aplicarlos con éxito.

15 Bibliografía y recursos

[ACHO13] Christophe Achouiantz. Depth of Kanban – A Good Coaching Tool. [URL](#).

[ANDERS10] David J. Anderson. Kanban: Successful Evolutionary Change for Your Technology Business

[ANDERS12] David J. Anderson y otros. How deep is your Kanban? Accesible en la siguiente [URL](#).

[BALSQ] Balsamiq. Herramienta de prototipado de interfaces. [URL](#).

[BAIER07] Christel Baier and Joost-Pieter Katoen. Principles of Model Checking.

[BARNARD] Alan Barnard. What is Theory of Constraints (TOC). [URL](#).

[BARR98] Michael Barr y Charles Wells. Category Theory for Computing Science. [URL](#).

[BECK04] Kent Beck. Extreme Programming Explained.

[BROWN14] Simon Brown. Software Architecture for Developers. [URL del libro](#).

[CITIC] Fundación CITIC. [URL](#).

[DARKPATT] Darkpatterns.org. [URL](#).

[DEMARCO99] Tom De Marco y Timothy Listener. Peopleware: Productive Projects and Teams.

[DISHMAN15] Molly Dishman y Martin Fowler. Agile Architecture Keynote. [URL](#).

[DJBDD] Diego J. Romero López. Biblioteca de Árboles Binarios de Decisión para representar estados software. [URL](#).

[DNF] Disjunctive normal form. Encyclopedia of Mathematics. [URL](#).

[FORSS12] Hakan Forss. Are the Kanban practices in the right order? [URL](#).

[FOWL99] Martin Fowler. Refactoring: Improving the Design of Existing Code.

[GARZAS16] Javier Garzás. Agilidad y Lean. Gestionando los proyectos y negocios del S. XXI (5^a edición). Curso en línea accesible en la siguiente [URL](#).

[KNUTH84] Donald E. Knuth. Literate programming. Disponible en el siguiente [enlace](#).

[KRUG14] Steve Krug. Don't make me think 3^a Ed..

[LEAN] James Womack, Daniel T. Jones y Daniel Roos. The Machine That Changed The World.

[LINDERS16] Ben Linders. Agile Self Assessments [URL](#)

[MCCON04] Steve McConnell. Code Complete 2: A practical handbook for software construction.

[MARTIN08] Robert C. Martin. Clean Code: A handbook of Agile Software Craftsmanship.

[MOSELEY06] Ben Moseley y Peter Marks. Out of the Tar Pit.

[NIELSEN93] Usability Engineering. Jakob Nielsen.

[OCAML] OCaml. Lenguaje de programación funcional. [URL](#).

[NIELSEN95] Ten usability heuristics. Jakob Nielsen. [URL](#).

[PAWSON04] Richard Pawson. Naked Objects. Tesis para la obtención del Doctorado en Filosofía en la Universidad Trinity College de Dublín (Irlanda). [URL](#).

[PARMIN2012] C. Parnin y S. Rugaber. Programmer Information Needs After Memory Failure. ICPC 2012.

[PERCIVAL15] Harry J. W. Percival. Test-Driven Development with Python.

[PYSTTRLO] PyStats-Trello. Generador de estadísticas para Trello. [URL](#).

[PYTRELLO] Py-Trello. Envoltura de la API de Trello. Versión con estadísticas [URL](#).

[RASM10] Jonathan Rasmunsson. The Agile Samurai.

[RIBEIRO05] Eduardo Ribeiro. Agile Maturity Self-Assessment Survey [URL](#).

[SCHIFFER11] Bernd Schiffer. How to Track the Team's Mood with a Niko-niko Calendar. [URL](#).

[TAWDIS] TAW Online. Evaluador de accesibilidad web en línea. Disponible en la siguiente [URL](#).

[TOTA11Y] Tota11y herramienta de evaluación de la accesibilidad. Kahn Academy.

[TRELLO] Trello. Herramienta web gratuita de implantación de Kanban. Disponible en la [siguiente URL](#).

[UBOK] Usability Body of Knowledge. En la siguiente [URL](#).

[VALPAK13] Where Are We In Our Agile Journey?: An Agile Survey Approach. [URL](#).

[WCAG20] Web Content Accessibility Guidelines (WCAG). W3C. [URL](#).

[WATERS08] Kelly Waters. How agile are you? [URL](#)

[WELIE] Welie.com pattern library. [URL](#).

[WEBUX] User Experience for the Web (WebUX). Curso *online* disponible en la siguiente [URL](#).

[YAHOODPL] Yahoo design pattern library. [URL](#).

16 Anexo 1: cuestionario de evaluación de implantación Kanban de Christophe Achouiantz

Visualize

1. Work (all, according to current policies)
2. Work Types
3. Workflow ("process", way-of-working, value stream)
4. 'Next' & 'Done'
5. Current Team Focus (avatars)
6. Blocks
7. Current Policies (DoD, DoR, capacity allocations, etc.)
8. Ready for Pull ("done" within the workflow/in columns)
9. Metrics (lead-times, local cycle times, SLA targets, etc.)
10. WIP limits
11. Inter-work dependencies (hierarchical, parent-child, etc.)
12. Inter-workflow dependencies
13. Risk dimensions (cost-of-delay, technical risk, market risk)

Make Policies Explicit - All Policies must be DOCUMENTED (written & publicly used)

1. Definition of Work Types and Work Item (template)
2. How to pull work (selection from 'Next' / prioritization of WIP)
3. Who and when manages the 'Next' and 'Done' queues
4. Staff allocation / work assignment (individual focus)
5. Definition of Ready for 'Next'
6. Who, when and how to estimate work size
7. Limit size of work items (work breakdown)
8. How to select & prepare work for the 'Next' queue
9. Definition of Done at all steps (seen as a Target Condition)
10. Knowledge spreading/sharing strategy
11. Class-of-Service
12. Capacity allocation

Effects (seeing Evidence of...)

1. Team members are seeing and understanding the Big Picture (team-level vs. local situations)
2. Better "team spirit" (helping each-others to complete work, respect)
3. Focus on removing blocks
4. Focusing on finishing work rather than starting new work
5. Team is working on the "right" thing ("right" prioritization)
6. Limiting work to team's capacity (limited stress, optimal lead-times)
7. Team has motivation to drive improvements
8. Local process evolution (visualization, workflow, policies, WIP limits)
9. Increase depth of Kanban implementation
10. Process evolution was model-driven
11. Process or management policy evolution as a result of mentor-mentee
12. Inter-workflow or management policy evolution due to operations review

Depth of Kanban Implementation

Team:

Date:

Limit Work in Progress

1. No WIP limit, but commitment to finishing work over starting new (eventually reaching a WIP level that "feels OK" for the team)
2. Some explicit WIP limits, at lower level than workflow (a.k.a Proto-Kanban): personal Kanban, WIP limit per person, WIP limits for some columns or swim-lanes, workflow with infinite limits on "done" queues, etc.
3. Explicit WIP limit at workflow level - Single workflow full pull
4. Multiple interdependent workflows with pull system

Manage Flow

1. Daily planning meetings (as "daily" as agreed by policies)
2. Blocks out of team control are escalated for resolution
3. Next is re-prioritized continuously (no commitment in Next)- Deferred Pull decisions (dynamic prioritization)
4. CFDs (updated at least once a week)
5. Control Charts (updated at least once a week)
6. Size of ongoing work items is limited (large work is broken down)
7. Flexible staff allocation (swarming)
8. Cadence is established (planning, delivering, retrospective)
9. Flow metrics (number of days blocked, lead-time efficiency)
10. SLA expectations and forecasts (lead-time targets)
11. Capacity Allocations

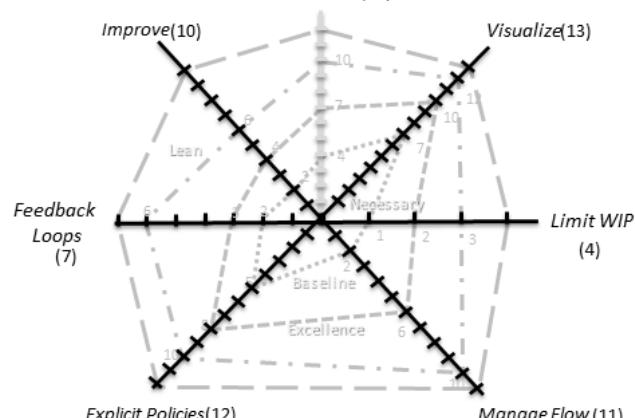
Implement Feedback Loops

1. Daily Team standups
2. Key stakeholders (mngt, customers, other groups) are regularly updated on the current situation
3. Managers go and see (walk the 'gemba') regularly
4. Regular discussions with upstream and downstream partners
5. Regular presentations and discussions about Financial performance
6. Regular presentations and discussions about Quality KPI (defect rate, customer satisfaction, etc.)
7. "Regularly" means once per month or more often

Improve

1. The group knows why it exists and its criteria for success
2. Regular Retrospectives / Kaizen events
3. The team knows its current condition (may require metrics)
4. True North exists, is communicated and shared by the team
5. The team knows the current target condition (the challenge)
6. There is a validation criteria (test) for the current target condition to know when the target condition is reached
7. The team knows what obstacles are preventing them from reaching the target condition
8. The team knows what obstacle is being currently addressed
9. The team knows what is the next step in resolving the current obstacle (PDCA)
10. The team go and see what they have learned from taking that step

Effects (12)



17 Anexo 2: manual de usuario y resultados de ejemplo de PyStats-Trello

17.1 Fichero Readme

A continuación se muestre el fichero README.md del repositorio PyStats-Trello. Este fichero indica cómo se ha de instalar el software y cómo usarlo.

Pystats-trello

Statistics and charts for Trello boards.

These small utilities gives you the functionality needed to extract some metric from Trello Kanban boards.

Requirements

See requirements.txt.

Implemented Kanban Metrics

Time by column

Average time the cards spend in each column.

Very useful to detect bottlenecks in your management process or project.

Forward and backward movements per column

Number of forward movements and bacward movements that have a column as a source.

Cycle

Time between development state and reaching "Done" state.

The average development and deployment time for all tasks of board.

Lead time

Time from start to end ("Done" state).

Time a client has to wait to see a feature he/she asked.

Time each card has been in each column

Based on movement operations, it is computed the time each card is in each column.

Spent and estimated times for each card

Some plugins like [\[Plus for Trello\]](http://www.plusfortrello.com/p/about.html)(<http://www.plusfortrello.com/p/about.html>) allow the insertion of spent and estimated time values in the comments of each card.

Specifying a regex in the settings_local allow the system to fetch them and

interpret them.

Please, note that we have supposed that if there are several comments with different

numbers, the numbers must be added. This is the case of Plus for Trello but your case could be different.

Installation

Create a virtualenv and install requirements.txt there.

```
```shell
$ virtualenv venv
```

```

Activate the **virtualenv** and follow next steps.

```
```shell
$. venv/bin/activate
```

```

Install requirements

```
```shell
(venv)$ pip install -r requirements
```

```

That's all, now configure your API keys and configure your board.

Configuration

First you have to create a configuration file in the root of the project with the name **settings_local.py**.

This file will board preferences:

```
```txt
BOARD_NAME: <BOARD NAME>
DEVELOPMENT_LIST: <DEFAULT_DEVELOPMENT_LIST_NAME>
DONE_LIST: <DEFAULT_DONE_LIST_NAME>
CUSTOM_WORKFLOWS:
- <CUSTOM_WORKFLOW_1_NAME>
 - LISTS: <LIST_1A_NAME>, <LIST_1B_NAME>, ... <LIST_1Z_NAME>
 - DONE_LISTS: ... <LIST_1Z_NAME>
- <CUSTOM_WORKFLOW_2_NAME>
 - LISTS: <LIST_2A_NAME>, <LIST_2B_NAME>, ... <LIST_2Z_NAME>
 - DONE_LISTS: ... <LIST_2Z_NAME>
CARD_ACTION_FILTER: [YYYY-MM-DD, YYYY-MM-DD]
CARD_IS_ACTIVE_FUNCTION: <CARD IS ACTIVE CONDITION> (by default is not card.closed)
COMMENT_SPENT_ESTIMATED_TIME_REGEX: PLUS_FOR_TRELLO_REGEX
OUTPUT_DIR: <OUTPUT DIR>
```

```

Configuration example

```
```txt
BOARD_NAME: My Tasks
DEVELOPMENT_LIST: Development
```

```

```

DONE_LIST: Done
CUSTOM_WORKFLOWS:
- Tasks waiting pass to deployment server
  - LISTS: Pending pass a deployment server
  - DONE_LISTS: Done
CARD_ACTION_FILTER: [2016-03-01, 2016-04-01]
CARD_IS_ACTIVE_FUNCTION: not card.closed
COMMENT_SPENT_ESTIMATED_TIME_REGEX: PLUS_FOR_TRELLO_REGEX
OUTPUT_DIR: ./results/my-tasks
```

```

How to use it

```

```shell
(venv)$ python stats_extractor.py <configuration_file>
Where:
- <configuration_file> is the file path of the configuration file with your preferences.
```

```

returns a summary of stats of the board\_name.

## 17.2 Fichero de salida de ejemplo

Fichero de salida de PyStats-Trello de uno de nuestros tableros (se han ocultado los nombres de las tareas y del tablero por motivos de privacidad).

```
Measurements for Board
```

```
General measurements for Board
```

- The board is 45895.1687833 hours old
- Last card was created 220.19727312 hours ago
- There are 38 tasks (37 active / 1 inactive [see note 1]) (
- There are 3 tasks in 'done' (0 are inactive [see note 1])
- 0.00392198143665 tasks per day or 6.53663572774e-05 tasks per hour

[note 1]: A card is active if meets this criterion: not card.closed

```
Average time in each column for all the board cards between dates 2016-04-01
and 2016-04-20
```

- Pendientes de estimación: 113.72 h (std. dev. 154.23)
- Estimadas: 55.59 h (std. dev. 131.88)
- Pendientes de desarrollo: 23.03 h (std. dev. 88.11)
- En desarrollo: 12.27 h (std. dev. 69.35)
- En revisión técnica: 10.08 h (std. dev. 35.89)
- Pendiente de subida: 0.46 h (std. dev. 1.91)

- En revisión comercial: 243.86 h (std. dev. 297.52)
- Terminadas: 37.19 h (std. dev. 157.11)

```
Sum of forward/backward movements by source column for all the cards between dates 2016-04-01 and 2016-04-20
```

- Pendientes de estimación
  - Forward movements: 14
  - Backward movements: 0
- Estimadas
  - Forward movements: 3
  - Backward movements: 1
- Pendientes de desarrollo
  - Forward movements: 6
  - Backward movements: 1
- En desarrollo
  - Forward movements: 7
  - Backward movements: 1
- En revisión técnica
  - Forward movements: 5
  - Backward movements: 0
- Pendiente de subida
  - Forward movements: 6
  - Backward movements: 0
- En revisión comercial
  - Forward movements: 0
  - Backward movements: 4
- Terminadas
  - Forward movements: 0
  - Backward movements: 0

```
Forward/backward movements movements by username in this board between dates 2016-04-01 and 2016-04-20
```

- Forward movements of User 56ebe77e29beca6cae30ec0f's tasks: 0

- Backward movements of User 56ebe77e29beca6cae30ec0f's tasks: 0
- Forward movements of User 56bc6cf6e157e9d459127712's tasks: 0
- Backward movements of User 56bc6cf6e157e9d459127712's tasks: 1
- Forward movements of User 56ebeaf7df0bc44c725fc0e2's tasks: 0
- Backward movements of User 56ebeaf7df0bc44c725fc0e2's tasks: 0
- Forward movements of User 56e2ac8e14e4eda06ac6b8fd's tasks: 10
- Backward movements of User 56e2ac8e14e4eda06ac6b8fd's tasks: 2
- Forward movements of User 56ebe95058327a2bd6564ce3's tasks: 10
- Backward movements of User 56ebe95058327a2bd6564ce3's tasks: 3

## ## Cycle

Time between development state and reaching 'Done' state between dates 2016-04-01 and 2016-04-20

- 570f892becdf63e5b31c965e Task 570f892becdf63e5b31c965e: 0.00
- 56eff9ff504beeb14f2e1be2 Task 56eff9ff504beeb14f2e1be2: 781.69
- 56fa6eaf7410e874b91c2403 Task 56fa6eaf7410e874b91c2403: 594.47
- avg: 458.72 h, std\_dev: 333.25

## ## Lead

Time from start to end ('Done' state) between dates 2016-04-01 and 2016-04-20

- 570f892becdf63e5b31c965e Task 570f892becdf63e5b31c965e: 116.33
- 56eff9ff504beeb14f2e1be2 Task 56eff9ff504beeb14f2e1be2: 781.69
- 56fa6eaf7410e874b91c2403 Task 56fa6eaf7410e874b91c2403: 594.47
- avg: 497.49 h, std\_dev: 280.15

## ## Custom workflow Desarrollo Tareas CDR

- 570bc2bcecbe26154db636d7 'Task 570bc2bcecbe26154db636d7': 161.13
- 570b624dd060a43e11ee452d 'Task 570b624dd060a43e11ee452d': 3.57
- 56fe65e6e083b119fa34f420 'Task 56fe65e6e083b119fa34f420': 126.90
- 56fb9b74468ae6fcf1d8c49d 'Task 56fb9b74468ae6fcf1d8c49d': 127.14
- 56fba98be1ae87a8f51b4116 'Task 56fba98be1ae87a8f51b4116': 116.21
- 56ec039cabcc2221d63066a54 'Task 56ec039cabcc2221d63066a54': 443.89
- 56fa2831ed8ad92872404a6a 'Task 56fa2831ed8ad92872404a6a': 172.17

```

- 57035f693458de383bdcd4b5 'Task 57035f693458de383bdcd4b5': 0.48
- 56ec039426e16e6d726ec89e 'Task 56ec039426e16e6d726ec89e': 0.00
- 56ec03c72a844cb51a0dca2b 'Task 56ec03c72a844cb51a0dca2b': 0.00
- 56eff41f687edfeecd6089a 'Task 56eff41f687edfeecd6089a': 0.00
- 56f0fb1379e017b3c6a1460e 'Task 56f0fb1379e017b3c6a1460e': 0.00
- 56fb993d9074b813950dcefe 'Task 56fb993d9074b813950dcefe': 0.00
- 56fd3573243ccb7d15071974 'Task 56fd3573243ccb7d15071974': 0.00
- 56fd3568edb52ec85ac61490 'Task 56fd3568edb52ec85ac61490': 21.04
- 56f1c4a98f47ab169e452a5e 'Task 56f1c4a98f47ab169e452a5e': 0.00
- 56fe8b8871024ee51e29e3b0 'Task 56fe8b8871024ee51e29e3b0': 0.00
- 570f892becdf63e5b31c965e 'Task 570f892becdf63e5b31c965e': 0.00
- 56eff9ff504beeb14f2e1be2 'Task 56eff9ff504beeb14f2e1be2': 0.00
- 56fa6eaf7410e874b91c2403 'Task 56fa6eaf7410e874b91c2403': 0.00
- avg: 58.63 h, std_dev: 107.27

```

#### ## Custom workflow Subidas Tareas CDR

```

- 570f892becdf63e5b31c965e 'Task 570f892becdf63e5b31c965e': 0.00
- 56eff9ff504beeb14f2e1be2 'Task 56eff9ff504beeb14f2e1be2': 0.00
- 56fa6eaf7410e874b91c2403 'Task 56fa6eaf7410e874b91c2403': 0.00
- avg: 0.00 h, std_dev: 0.00

```

## Time each card has been in each column (hours) between dates 2016-04-01 and 2016-04-20

Card\_id Card\_name Pendientes de estimación, Estimadas, Pendientes de desarrollo, En desarrollo, En revisión técnica, Pendiente de subida, En revisión comercial, Terminadas

```

- 56fd3bd2bd64d2de4bd53dae 'Task 56fd3bd2bd64d2de4bd53dae': 544.29, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 56fcf4cb72471bed926f164a 'Task 56fcf4cb72471bed926f164a': 549.26, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 5707a70c940c265141a90617 'Task 5707a70c940c265141a90617': 357.73, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570b64afc608658ca9c9a97f 'Task 570b64afc608658ca9c9a97f': 290.75, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570bac09fc4f0d14ac6eaf15 'Task 570bac09fc4f0d14ac6eaf15': 285.76, 0.00, 0.00,

```

0.00, 0.00, 0.00, 0.00, 0.00

- 570cb33ffb2beeca4c56d0b1 'Task 570cb33ffb2beeca4c56d0b1': 267.35, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570cf9496b86c3e19db62b36 'Task 570cf9496b86c3e19db62b36': 262.45, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570e192f2e5b63f94fb24c5e 'Task 570e192f2e5b63f94fb24c5e': 242.32, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570f6a70f4bf7dc0c54eee2b 'Task 570f6a70f4bf7dc0c54eee2b': 218.73, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 570f6b41be9714711e3fcb63 'Task 570f6b41be9714711e3fcb63': 218.67, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 56fe663583f0d815c4f7a277 'Task 56fe663583f0d815c4f7a277': 269.06, 254.37, 0.00, 0.00, 0.00, 0.00, 0.00
- 56f01aa38d7c0830c3c0eb01 'Task 56f01aa38d7c0830c3c0eb01': 255.72, 523.68, 0.00, 0.00, 0.00, 0.00, 0.00
- 5704e0443b0315ae531dd19a 'Task 5704e0443b0315ae531dd19a': 0.00, 407.31, 0.14, 0.00, 0.00, 0.00, 0.00, 0.00
- 57068b41cb099c62b99ae001 'Task 57068b41cb099c62b99ae001': 86.99, 290.59, 0.00, 0.00, 0.00, 0.00, 0.00
- 57077dd9f05d9ebaa897e3ce 'Task 57077dd9f05d9ebaa897e3ce': 70.14, 290.48, 0.00, 0.00, 0.00, 0.00, 0.00
- 570b6da2716d17856a08376f 'Task 570b6da2716d17856a08376f': 0.01, 290.12, 0.00, 0.00, 0.00, 0.00, 0.00
- 56fb7763a93986e4cc6844db 'Task 56fb7763a93986e4cc6844db': 52.26, 0.00, 523.68, 0.00, 0.00, 0.00, 0.00, 0.00
- 570bc2bcecbe26154db636d7 'Task 570bc2bcecbe26154db636d7': 0.22, 0.00, 17.05, 0.28, 143.80, 0.00, 122.82, 0.00
- 570b624dd060a43e11ee452d 'Task 570b624dd060a43e11ee452d': 0.24, 0.16, 0.96, 2.60, 0.00, 0.00, 286.96, 0.00
- 56fe65e6e083b119fa34f420 'Task 56fe65e6e083b119fa34f420': 0.01, 0.00, 97.48, 1.30, 19.70, 8.43, 396.54, 0.00
- 56fb9b74468ae6fcf1d8c49d 'Task 56fb9b74468ae6fcf1d8c49d': 49.74, 0.00, 96.66, 0.58, 21.48, 8.43, 396.54, 0.00
- 56fba98be1ae87a8f51b4116 'Task 56fba98be1ae87a8f51b4116': 48.76, 0.00, 116.21, 0.00, 0.00, 0.00, 407.47, 0.00
- 56ec039cabcc2221d63066a54 'Task 56ec039cabcc2221d63066a54': 0.00, 0.00, 0.00, 427.90, 15.99, 0.00, 408.75, 0.00
- 56fa2831ed8ad92872404a6a 'Task 56fa2831ed8ad92872404a6a': 0.00, 0.00, 0.00, 0.00, 172.17, 0.00, 427.22, 0.00

- 57035f693458de383bdcd4b5 'Task 57035f693458de383bdcd4b5': 0.00, 0.00, 0.00, 0.48, 0.00, 0.00, 433.90, 0.00
- 56ec039426e16e6d726ec89e 'Task 56ec039426e16e6d726ec89e': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 852.64, 0.00
- 56ec03c72a844cb51a0dca2b 'Task 56ec03c72a844cb51a0dca2b': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 852.62, 0.00
- 56eff41f687edfeecdb6089a 'Task 56eff41f687edfeecdb6089a': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 782.10, 0.00
- 56f0fb1379e017b3c6a1460e 'Task 56f0fb1379e017b3c6a1460e': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 763.70, 0.00
- 56fb993d9074b813950dcefe 'Task 56fb993d9074b813950dcefe': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 573.58, 0.00
- 56fd3573243ccb7d15071974 'Task 56fd3573243ccb7d15071974': 21.03, 0.00, 0.00, 0.00, 0.00, 0.00, 523.72, 0.00
- 56fd3568edb52ec85ac61490 'Task 56fd3568edb52ec85ac61490': 0.00, 0.00, 0.00, 21.04, 0.00, 0.00, 523.71, 0.00
- 56f1c4a98f47ab169e452a5e 'Task 56f1c4a98f47ab169e452a5e': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 749.60, 0.00
- 56fe8b8871024ee51e29e3b0 'Task 56fe8b8871024ee51e29e3b0': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 520.82, 0.00
- 570f892becdf63e5b31c965e 'Task 570f892becdf63e5b31c965e': 116.33, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
- 56eff9ff504beeb14f2e1be2 'Task 56eff9ff504beeb14f2e1be2': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 781.69
- 56fa6eaf7410e874b91c2403 'Task 56fa6eaf7410e874b91c2403': 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 594.47

```
Total spent and estimated times for each card (in units given by plugin)
Card_id Card_name CurrentList Spent Estimated
- 56fd3bd2bd64d2de4bd53dae 'Task 56fd3bd2bd64d2de4bd53dae' (Pendientes de estimación): N/A N/A
- 56fcf4cb72471bed926f164a 'Task 56fcf4cb72471bed926f164a' (Pendientes de estimación): N/A N/A
- 5707a70c940c265141a90617 'Task 5707a70c940c265141a90617' (Pendientes de estimación): N/A N/A
- 570b64afc608658ca9c9a97f 'Task 570b64afc608658ca9c9a97f' (Pendientes de estimación): N/A N/A
- 570bac09fc4f0d14ac6eaf15 'Task 570bac09fc4f0d14ac6eaf15' (Pendientes de estimación): N/A N/A
```

- 570cb33ffb2beeca4c56d0b1 'Task 570cb33ffb2beeca4c56d0b1' (Pendientes de estimación): N/A N/A
- 570cf9496b86c3e19db62b36 'Task 570cf9496b86c3e19db62b36' (Pendientes de estimación): N/A N/A
- 570e192f2e5b63f94fb24c5e 'Task 570e192f2e5b63f94fb24c5e' (Pendientes de estimación): N/A N/A
- 570f6a70f4bf7dc0c54eee2b 'Task 570f6a70f4bf7dc0c54eee2b' (Pendientes de estimación): N/A N/A
- 570f6b41be9714711e3fc63 'Task 570f6b41be9714711e3fc63' (Pendientes de estimación): N/A N/A
- 56fe663583f0d815c4f7a277 'Task 56fe663583f0d815c4f7a277' (Estimadas): N/A N/A
- 56f01aa38d7c0830c3c0eb01 'Task 56f01aa38d7c0830c3c0eb01' (Estimadas): N/A N/A
- 5704e0443b0315ae531dd19a 'Task 5704e0443b0315ae531dd19a' (Estimadas): N/A N/A
- 57068b41cb099c62b99ae001 'Task 57068b41cb099c62b99ae001' (Estimadas): N/A N/A
- 57077dd9f05d9ebaa897e3ce 'Task 57077dd9f05d9ebaa897e3ce' (Estimadas): N/A N/A
- 570b6da2716d17856a08376f 'Task 570b6da2716d17856a08376f' (Estimadas): N/A N/A
- 56fb7763a93986e4cc6844db 'Task 56fb7763a93986e4cc6844db' (Pendientes de desarrollo): N/A N/A
- 570bc2bcecbcbe26154db636d7 'Task 570bc2bcecbcbe26154db636d7' (En revisión comercial): 0.5 0.5
- 570b624dd060a43e11ee452d 'Task 570b624dd060a43e11ee452d' (En revisión comercial): N/A N/A
- 56fe65e6e083b119fa34f420 'Task 56fe65e6e083b119fa34f420' (En revisión comercial): 1.45 1.3
- 56fb9b74468ae6fcf1d8c49d 'Task 56fb9b74468ae6fcf1d8c49d' (En revisión comercial): 2.5 2.5
- 56fba98be1ae87a8f51b4116 'Task 56fba98be1ae87a8f51b4116' (En revisión comercial): 1.0 4.0
- 56ec039cabcc2221d63066a54 'Task 56ec039cabcc2221d63066a54' (En revisión comercial): N/A N/A
- 56fa2831ed8ad92872404a6a 'Task 56fa2831ed8ad92872404a6a' (En revisión comercial): N/A N/A
- 57035f693458de383bdcd4b5 'Task 57035f693458de383bdcd4b5' (En revisión comercial): N/A N/A
- 56ec039426e16e6d726ec89e 'Task 56ec039426e16e6d726ec89e' (En revisión comercial): N/A N/A
- 56ec03c72a844cb51a0dca2b 'Task 56ec03c72a844cb51a0dca2b' (En revisión comercial): N/A N/A

- 56eff41f687edfeecdb6089a 'Task 56eff41f687edfeecdb6089a' (En revisión comercial): N/A N/A
- 56f0fb1379e017b3c6a1460e 'Task 56f0fb1379e017b3c6a1460e' (En revisión comercial): N/A N/A
- 56fb993d9074b813950dcefe 'Task 56fb993d9074b813950dcefe' (En revisión comercial): N/A N/A
- 56fd3573243ccb7d15071974 'Task 56fd3573243ccb7d15071974' (En revisión comercial): N/A N/A
- 56fd3568edb52ec85ac61490 'Task 56fd3568edb52ec85ac61490' (En revisión comercial): N/A N/A
- 56f1c4a98f47ab169e452a5e 'Task 56f1c4a98f47ab169e452a5e' (En revisión comercial): N/A N/A
- 56fe8b8871024ee51e29e3b0 'Task 56fe8b8871024ee51e29e3b0' (En revisión comercial): N/A N/A
- 570f892becdf63e5b31c965e 'Task 570f892becdf63e5b31c965e' (Terminadas): N/A N/A
- 56eff9ff504beeb14f2e1be2 'Task 56eff9ff504beeb14f2e1be2' (Terminadas): N/A N/A
- 56fa6eaf7410e874b91c2403 'Task 56fa6eaf7410e874b91c2403' (Terminadas): N/A N/A
- Spent Times avg: 1.36 h, std\_dev: 0.74
- Estimated Times avg: 2.08 h, std\_dev: 1.32

### Total spent/estimated times for each user per MONTH (in units given by plugin)

- User 56e2ac8e14e4eda06ac6b8fd
  - 2016-M04: 5.45 / 8.30 (diff. -2.85)
- User 56bc6cf6e157e9d459127712
- User 56ebe77e29beca6cae30ec0f
- User 56ebeaf7df0bc44c725fc0e2
- User 56ebe95058327a2bd6564ce3

### Total spent/estimated times for each user per WEEK (in units given by plugin)

- User 56e2ac8e14e4eda06ac6b8fd
  - 2016-W14: 4.95 / 7.80 (diff. -2.85)
  - 2016-W15: 0.50 / 0.50 (diff. 0.00)
- User 56bc6cf6e157e9d459127712
- User 56ebe77e29beca6cae30ec0f

- User 56ebeaf7df0bc44c725fc0e2
- User 56ebe95058327a2bd6564ce3

```
Number of cards created by month for each label per MONTH
```

- 2016-M03
  - Label 56fe625f56197bf12dd1484b: 1
  - Label 56ebe73c152c3f92fd79863d: 2
  - Label 56f91d8c152c3f92fd96a307: 3
- 2016-M04
  - Label 56fe625f56197bf12dd1484b: 1
  - Label 56ebe73c152c3f92fd79863d: 1

```
Number of cards created by week for each label per WEEK
```

- 2016-W11
- 2016-W12
  - Label 56f91d8c152c3f92fd96a307: 1
- 2016-W13
  - Label 56fe625f56197bf12dd1484b: 2
  - Label 56ebe73c152c3f92fd79863d: 3
  - Label 56f91d8c152c3f92fd96a307: 2
- 2016-W14
- 2016-W15

Charts done

--- END OF FILE ---