UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SOFTWARE COMPLEMENTARIO PARA TABLERO KANBAN FÍSICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

SERGIO ESTEBAN POLA CONTRERAS

PROFESOR GUIA:
AGUSTÍN ANTONIO VILLENA MOYA

MIEMBROS DE LA COMISIÓN:
SERGIO OCHOA DELORENZI
PABLO ANTONIO GONZÁLEZ JURE

SANTIAGO DE CHILE
JUNIO 2012

Desde hace ya un par de décadas, las metodologías ágiles o *lean thinking*, han revolucionado la forma de trabajar, no sólo en ingeniería de software, sino en gran parte de la industria. Donde los principios básicos apuntan al trabajo en equipo, al aportar valor al proyecto cuanto antes y a la capacitación de desarrolladores. Los procesos se flexibilizan y se vuelven transversales, opuesto al ordenamiento en cascada, lo que hasta ese entonces se realizaba.

Kanban se puede considerar como una herramienta de apoyo a las metodologías ágiles. Bajo el simple objetivo de mantener la información visible, *Kanban* se ha convertido en un excelente complemento para quienes utilizan las metodologías ágiles, aportando a la flexibilidad de los procesos y al trabajo en equipo.

Pero tiene ciertas falencias típicas de lo que vulgarmente sería un tablero con papeles de colores: se puede caer y perder toda la información que contenía, no hay como incluir automáticamente indicadores de gestión, etcétera. Este Trabajo de Título pretende corregir las falencias de mayor impacto en el desarrollo a través de una aplicación que actúe como respaldo del *Kanban*, donde las tareas sean ingresadas junto a sus estados y se pueda verificar, por ejemplo la ultima configuración conocida de la misma.

El desarrollo de la aplicación consistió en cuatro etapas, cada una de las cuales entrega una versión funcional de la aplicación. A medida que se avanzaba se iba revisando las falencias para poder mejorarlas en la siguiente versión, siendo la interfaz gráfica donde se realizó la mayor cantidad de mejoras.

Como casos de prueba, la versión final de la aplicación se entregó a tres empresas, dos de ellas sin experiencia previa en agilidad, quienes la utilizaron durante tres meses, reportando fallos y posibles mejoras. Esto sirvió además para estudiar la adaptación de cada empresa a la aplicación.

Los resultados que se obtuvieron fueron satisfactorios, tanto en el desarrollo, funcionamiento y casos de prueba.

INDICE GENERAL

1 Introducción

Este Trabajo de Título aborda una de las herramientas o metodologías más importantes de la agilidad y del *lean thinking*: el *Kanban*. El punto de vista propuesto implica el desarrollo de una herramienta para suplir las desventajas del *Kanban* como herramienta física, en un formato digital, dándole ventajas adicionales de otro tipo de herramientas, pero manteniendo la flexibilidad por la que se caracteriza.

Para poner en contexto la importancia de esta herramienta o metodología en las metodologías ágiles de desarrollo tanto de productos como de software, como marco teórico, se revisa la historia del *lean thinking* y como el *Kanban* se ha abierto paso incluso entre los menos crédulos y ha pasado de ser una "pizarra con papelitos" a una de las herramientas más importantes y destacadas en la cultura ágil de desarrollo.

Este Trabajo de Título no cuestiona la importancia del *Kanban*, si no que se enfoca en sus debilidades, que como cualquier herramienta física posee. Por lo tanto, el objetivo general es mejorar el *Kanban*, a través de una herramienta digital complementaria, que permita tener un *back up* de la herramienta física, mientras calcula ciertos indicadores. Así es como se busca potenciar el *Kanban* y su uso en agilidad, generando un valor agregado a éste.

El Trabajo de Título, se enfoca en el desarrollo de la herramienta complementaria, sus características y los ejemplos de implementación que se han hecho hasta el momento.

1.1 Objetivos

1.1.1 Objetivo General

El objetivo de este Trabajo de Título consiste en desarrollar una aplicación que sea una compañera al *Kanban* físico supliendo sus carencias más relevantes como la falta de respaldo, historial, cálculo automático de indicadores y generación de reportes.

La facilidad de sincronización entre el *Kanban* físico y el *Kanban* virtual es vital para el éxito de este software, ya que si la información con que cuenta el *Kanban* virtual para realizar los cálculos de

indicadores o generación de reportes no es la misma que posee el *Kanban* físico, entonces los reportes no serán representativos de la realidad del proyecto. Un mecanismo de sincronización complejo también puede producir que la herramienta quede en desuso.

1.1.2 Objetivos Específicos

- Hacer un modelo lógico de un tablero *Kanban* simple que permita la generación de los reportes requeridos
- Desarrollar una interfaz usuaria que permita sincronizar los cambios de estados de las tareas
- Desarrollar una interfaz usuaria para recuperar el estado de las tareas ingresadas en el sistema
- Determinar y calcular indicadores relevantes para el desarrollo

1.2 Funcionalidades de la herramienta

Este Trabajo de Título plantea el desarrollo de una aplicación que aborde las desventajas del *Kanban* físico, mencionadas anteriormente, permitiendo formalizarlo como herramienta de trabajo. La solución planteada apunta a:

- Respaldo del estado del *Kanban* y posterior restauración
- Generación de índices de gestión

1.2.1 Respaldo del estado del *Kanban* y posterior restauración

Tal como se ha mencionado, uno de los usos de la solución a desarrollar está orientado al respaldo de las tareas y sus estados en cierto instante. Uno de los mecanismos de respaldo de los tableros *Kanban* son las fotografías tomadas periódicamente. Un defecto de este mecanismo es la dificultad para encontrar el estado de una tarea en un instante dado obligando a los equipos a recorrer cada fotografía en busca de una tarea.

El objetivo de poder tener respaldo de las tareas y sus estados, es tener la opción de recuperar la última configuración y distribución disponible para poder volver a trabajar después de cualquier problema que se presente, como en el caso en que alguien pase a llevar el tablero y se caigan algunas tarjetas con tareas, en este caso la aplicación puede entregar el último estado ingresada de ésta. Por ejemplo, en el juego de ajedrez sería útil tener una manera de registrar la posición de las piezas

después de cada movimiento de manera compacta y que dicho registro permita poder reconstruir el último estado del tablero.

1.2.2 Generación de índices de gestión

El objetivo de calcular índices es analizar la información provista por el sistema con el fin de tomar decisiones que permitan corregir problemas o reforzar aciertos. La comparación de indicadores obtenidos en distintas fechas sirve para medir la eficiencia e impacto de las medidas y decisiones tomadas, permitiendo hacer mejoras a los procesos al instante.

1.2.3 Disciplina de sincronización

En paralelo al desarrollo de la herramienta se debe desarrollar una cultura sobre el buen uso del *Kanban*. Esto significa que todo lo que la persona cambie en el tablero *Kanban* debe cambiarlo en el sistema de apoyo.

2 Marco teórico

2.1 Agilidad

Muchas veces los procesos productivos se ven atorados en procedimientos y estructuras de trabajo, tan estrictas y limitadas, que a pesar de que se trabaje de la mejor forma posible, no se puede evitar la ineficiencia, pérdida de recursos, redundancia, etc.

Esta ineficiencia, amarrada a los procesos sobre estructurados empeoraba a medida que las empresas eran exigidas con múltiples proyectos de diferentes envergaduras. Todo esto llevó por primera vez a una empresa (Toyota) a cambiar esta estructuración de manera radical, "agilizándola", creando la metodología JIT (*Just in Time*) donde sólo se producía lo demandado, evitando uso de bodegas, pérdida de recursos o sobre stock por inventario, con un férreo compromiso de los proveedores, haciendo participar a los trabajadores como equipos y no solo como parte de una línea productiva.

Toyota fue sólo el inicio para el *Lean Thinking* en la industria en general, traspasando hasta el área de desarrollo de software, lo que se tradujo en las actuales metodologías ágiles.

Las metodologías ágiles nacen en la década de los 90, dentro de las cuales destacan *Extreme Programming*, *Scrum*, *Open Unified Process*, *Kanban*, entre otras.

Estas metodologías se concentran en reforzar el trabajo en equipo, fomentar la capacitación y el constante aprendizaje de los desarrolladores, propiciando un buen ambiente de trabajo.

Además, las metodologías ágiles incitan a mantener una fluida comunicación e interacción con el cliente, para lograr un desarrollo eficiente, generando valor lo antes posible al proyecto, disminuyendo los riesgos técnicos, eliminando tiempos muertos y cumpliendo con los plazos establecidos.

A continuación se describen algunas prácticas y conceptos utilizados en las metodologías ágiles.

2.1.1 Stand Up Meeting

La práctica de *stand up meeting*, del inglés reunión de pie, tiene como objetivo el compartir el trabajo realizado por los miembros del equipo la jornada previa y la planificación de la jornada venidera. Esta reunión tiene que ser breve y resumida, por este motivo se realiza de pie cosa que si se alarga mucho los participantes se cansen y digan sólo lo importante. En resumen, es una actualización del estado de las tareas que se desarrollan.

Cada miembro debe decir lo que realizó la jornada anterior, lo que realizará la jornada actual, que es lo que evitó cumplir sus objetivos y como fue resuelto. A partir de lo que se dice que se va a realizar en la jornada, las tareas son tomadas por quien se comprometió a realizarlas y llevadas a cabo.

Esta actividad, que es clave para lograr el éxito en desarrollos ágiles, pese a que no está directamente relacionada con el *Kanban*, puede apoyar entregándole información importante. Para facilitar el *stand up meeting* la aplicación debería proveer una funcionalidad que permita obtener las tareas que hayan sido modificadas en un período de tiempo.

La aproximación anterior puede ser mejorada al tomar como foco el flujo de valor y considerar la premisa de "para comenzar comienza a terminar". En este caso los miembros del equipo discuten sobre las tareas que están más cerca de poder terminarse con el objetivo de generar valor lo antes posible. La aplicación deberá contar con una funcionalidad que permita ordenar las tareas según su estado, de este modo, el equipo podrá ver fácilmente cuales son las que están prontas a ser terminadas.

2.1.2 Estados y Planning Game

Kanban tiene tres estados base con los que un equipo debe comenzar a trabajar. Los estados **Por Hacer**, **Haciendo** y **Hecho** es lo mínimo que se requiere para modelar un proceso productivo. Estos tres estados resultaron ser suficientes para limitar el flujo de trabajo y conocer la capacidad del equipo.

En agilidad existe una práctica conocida como "*Planning Game*" en la cual todos los involucrados en el proyecto discuten cuales serán las siguientes actividades a ser realizadas

basado en los requerimientos del cliente. Además, a estas actividades se les asigna un peso que permite determinar la cantidad de actividades que pueden ser realizadas en un periodo de tiempo establecido por acuerdo.

Todas las tareas se ingresan al estado Por Hacer donde el equipo le asigna un peso y a partir de ese peso el cliente, ya sea interno o externo, le asigna una prioridad.

- Asignación de peso: Para la asignación de peso se recurrió a la actividad conocida como *Planning Game* en la cual los requerimientos son analizados y desmenuzados en subtareas y éstas son detalladas. Cada miembro del equipo tiene en su poder un mazo de cartas con números que utilizan una escala elegida en el equipo
- Peso máximo: Si el peso de la tarea excede el límite definido por el equipo, entonces esta tarea debe ser dividida en subtareas y a las subtareas se les debe asignar el peso. El equipo debe tener presente que la suma de los pesos de las subtareas no debiera superar al peso de la tarea padre.
- Escala Numérica: La escala que utilizará el equipo para asignarle el peso a las tareas puede ser una escala de uno en uno, números primos o la serie de *Fibonacci* entre otras. La ventaja de la serie de *Fibonacci* por sobre una escala de uno en uno es que en la primera se hace visible la incertidumbre sobre una tarea a medida que esta va creciendo, por ejemplo, la tarea de peso inmediatamente superior a una de peso 8, es 13. De este modo, mientras más se alejan los pesos del origen más rápido se acercan al peso máximo forzando la subdivisión de la tarea y la discusión de ésta para lograr tareas pequeñas que resultan más sencillas de manejar.

Antes de comenzar el *Planning Game* se necesita definir quién será el moderador, que por lo general es el líder de equipo. Cada participante de la actividad debe poseer una mano de cartas con todos los pesos que pueden ser asignados a una tarea. Por cada tarea se realiza una ronda en la cual los participantes eligen el peso que le asignarán a la tarea y ponen la carta correspondiente volteada sobre la mesa.

2.2 *Kanban* Toyota [16]

Kanban es una herramienta de gestión inventada para señalar la necesidad de producir un componente dentro de una línea de producción. La palabra *Kanban* proviene del japonés donde kan, 看カン, significa "visual," y ban, 板バン, significa "tarjeta" o "tablero".

Un método de control único es como denominan en Toyota el *Kanban*, invención de ellos como parte del Sistema de Producción Toyota. También denominado el “Sistema Supermercado”, porque la idea principal fue tomada de estas tiendas comerciales. Estas tiendas de venta en masa utilizan tarjetas de control con información relacionada con el producto como el nombre, código, lugar de almacenaje, etc. Como Toyota tomó la idea y la implementó como *Kanban* en su línea de producción, así es como quedó el concepto. En Toyota, cuando un proceso se refiere a un proceso precedente para obtener piezas, se utilizan tarjetas *Kanban* para saber que piezas han sido usadas.

¿Por qué usar un concepto de supermercado? Un supermercado entrega sus productos a medida que los clientes van solicitándolos, en las cantidades que van solicitando y posee todos esos productos disponibles para ventas en cualquier momento.

Taiichi Ohno, quien promocionó la idea del *Just in Time*, aplicó este concepto creando una equivalencia del supermercado y sus clientes con los procesos precedentes y los procesos siguientes respectivamente. El próximo proceso era el equivalente al cliente y el proceso precedente el supermercado, donde se iba para obtener las partes necesarias cuando son necesarias en las cantidades que sean necesarias. De esta manera era posible mejorar el sistema ineficiente de producción que existía previamente. Ya no se fabricaban piezas en los procesos precedentes que no fueran solicitadas, evitando el sobre stock y su consecuente costo.

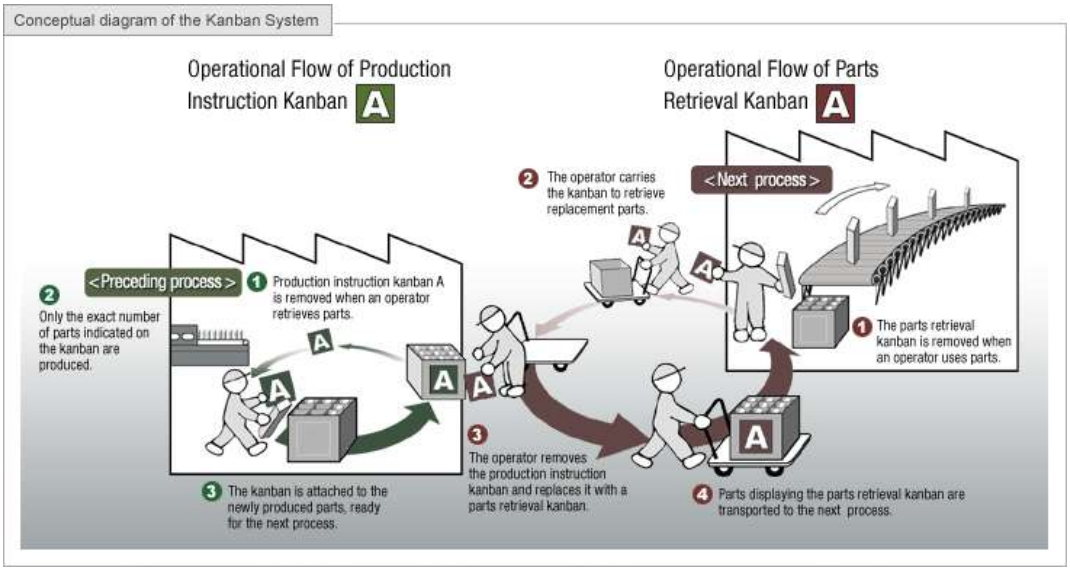

FIGURA 1. Diagrama de Producción de Toyota en base a *Kanban* [16]

2.3 *Kanban* en Desarrollo de Software

"*Kanban* para ingeniería de software" (*Kanban* a partir de ahora) es una metodología que recoge la experiencia del *Kanban* de Toyota y la lleva al mundo del desarrollo tecnológico siendo su expresión más común un tablero dividido en columnas que señalan los estados de un flujo de trabajo y tarjetas que van señalizando como fluyen los requerimientos dentro de un equipo de software.

Se basa en las siguientes propiedades fundamentales [4]:

- Visualizar el flujo de trabajo
- Limitar el trabajo en progreso
- Medir y manejar el flujo de trabajo
- Explicitar las políticas del proceso
- Usar modelos para mejorar oportunidades de mejora

FIGURA 2. Ejemplo de *Kanban* [9]

Kanban ha llegado a ser el exponente privilegiado de la gestión visual de información (*Visual Management*), cuyo objetivo es que la información vaya a las personas y no que las personas vayan a la información. Se les llama "radiadores de información" dado que las personas acceden a ésta sin que tengan que recurrir a repositorios de difícil acceso (Ejemplo: Carta Gantt de Microsoft Project).

A pesar del gran valor que entrega un tablero *Kanban* físico, adolece de algunos problemas tales como:

- Es difícil obtener métricas de gestión
- La información no es persistente, podría caerse un papel y no existe como saber donde se encontraba.
- El tablero físico sólo sirve para coordinar equipos ubicados en el mismo espacio de trabajo.

El objetivo de este Trabajo de Título es definir una metodología y una herramienta informática demostrativa de ésta que complemente el *Kanban* físico en estas áreas débiles

2.4 Características claves del *Kanban*

Existen algunas características claves que logran que se destaque en las herramientas de gestión:

a) Requiere poco aprendizaje para generar valor: La capacitación de una persona al uso del *Kanban* se reduce a enseñarle los códigos propios del equipo. Como en *Kanban* las políticas del proceso son explícitas el acceso y la comprensión de éstas se simplifica y disminuye los tiempos necesarios para que la persona entre en régimen.

b) Simple construcción y evolución: Al ser un tablero distribuido por zonas, armar un *Kanban* puede tomar menos de cinco minutos en su versión más simple [5]. Cambiar la distribución del tablero es sencillo y permite que ésta evolucione ajustándose a las necesidades que el equipo desee implementar en su flujo de trabajo. El tablero *Kanban* físico posee la flexibilidad necesaria para adaptarse a las necesidades del equipo que lo utilizará.

c) Baja inversión inicial y operacional: Construir un tablero *Kanban* resulta ser muy barato porque puede ser construido con materiales que se encuentran en cualquier oficina. Por ejemplo, una versión sencilla puede construirse sobre una muralla con cinta adhesiva de color para crear la distribución de zonas. Para operar un *Kanban* se necesita un lápiz y papeles de colores (*post-it)*, que son de bajo costo en el mercado.

d) Orientación a la generación de valor: La filosofía tras el *Kanban* sostiene que el flujo de las tareas debe ser en base a "jalarlas" de un estado a otro, concepto que se conoce como modelo *Pull*. Este modelo, junto a la priorización de las tareas basado en la cantidad de valor generado al cliente, garantizan que sólo se construirá lo que se necesite evitando la acumulación de tareas y el desperdicio de tiempo.

2.5 Desventajas del Tablero *Kanban* físico

Gran parte de las desventajas del tablero *Kanban* físico se producen por su carencia de no poder automatizar actividades. Entre las desventajas se encuentra la falta de historial de cambios, la capacidad de persistir la información y la carencia de generación de reportes e indicadores de manera rápida.

Medir y manejar el flujo se pueden convertir en una tarea ardua y tediosa. El cálculo de indicadores se va complicando a medida que el volumen de información que el *Kanban* provee crece o se requiere cambiar los indicadores o la forma en que se miden.

2.6 Tablero *Kanban* como soporte metodológico

Todo proceso de producción comprende una serie de actividades a ser realizadas. La existencia de dichas actividades es independiente del proceso o industria en la que se está inmerso, lo que permite la implementación del *Kanban* sin la necesidad de adaptar el proceso, con el objetivo de visualizar el estado de las actividades que comúnmente pasan inadvertidas generando problemas de falta de información en la toma de decisiones.

De esta manera, *Kanban* se convierte en un agente agilizador de procesos facilitando la visualización de los flujos de trabajo, los cuellos de botella existentes y actividades que están siendo sobre o subvaloradas, entre otras [6].

Por sus características, *Kanban* está siendo considerado como una meta-metodología. Esto implica que puede aplicarse sobre cualquier metodología que cuente con una organización, agilizando sus procesos [7]. Esto permite que la adopción del *Kanban* sea un proceso gradual minimizando el riesgo al rechazo por parte de la organización.

Tabla 1. Resumen Características

Kanban Físico	Kanban Virtual
Expresivo	Historial de Cambios
Radiador de Información	Persistir Información
Sencillo	Calculo automático de indicadores
Agente social	Accesible de varios sitios

2.7 Selección de Métricas

En los desarrollos ágiles los indicadores más relevantes son los siguientes:

- Tiempo de Ciclo: Es el tiempo que demora un requerimiento o tarea entre que es solicitado y es recibido con conformidad por el cliente

- Tiempo de Ingeniería: Es el tiempo que transcurre entre que el requerimiento es tomado por alguien y que es recibido con conformidad por el cliente.
- Tiempo de Redención: Es el tiempo que transcurre entre que un requerimiento es solicitado por el cliente y que es tomado por un desarrollador para empezar a trabajar en él.

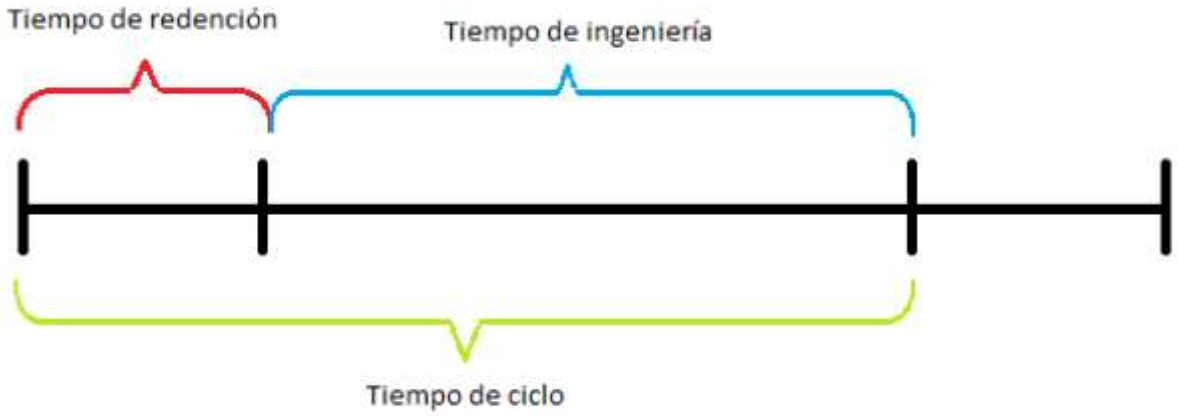

FIGURA 3. Diagrama de Métricas

En la Figura 3 se muestra gráficamente a que período corresponde cada métrica. La suma del tiempo de redención con el tiempo de ingeniería resulta en el tiempo de ciclo, es por esto es que el tiempo de ciclo resulta suficiente para comenzar a analizar el proceso de desarrollo de los equipos.

Estos indicadores tienen como fin, medir y controlar el flujo de trabajo para mantener el foco en la generación de valor temprana.

2.8 Ley de Little y diagrama de flujo acumulado

2.8.1 Gráfico "*Burn down*"

Uno de los objetivos de las metodologías ágiles en desarrollo de software, es aumentar la predictibilidad en la evolución de un desarrollo de un proyecto. Para esto, metodologías como *Scrum* usan diagramas como los gráficos "*Burn down*" que sirven para mostrar un estimado de las horas necesarias para terminar con las tareas pendientes de la iteración actual.

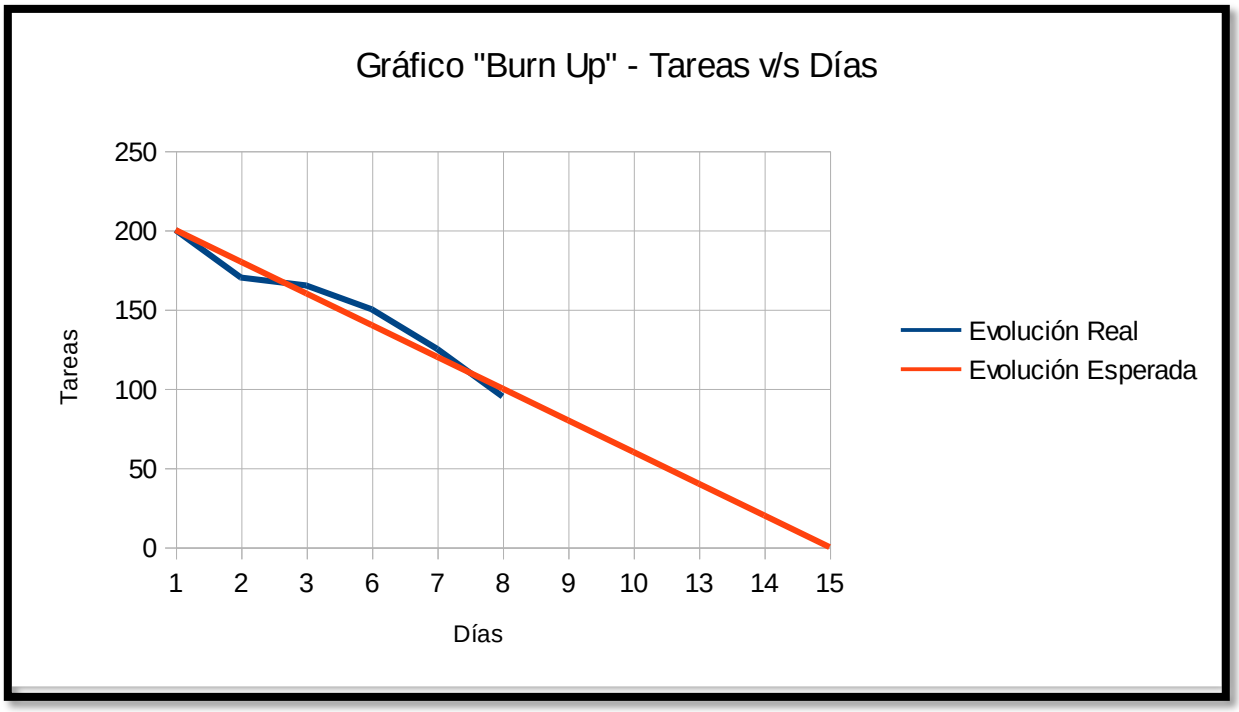

FIGURA 4 Gráfico "Burn Down"

El foco central de este gráfico son las horas de esfuerzo requeridas para terminar la iteración. La tendencia o evolución esperada se expresa con una línea que une la cantidad de horas remanentes iniciales en el eje vertical con la fecha de la entrega en el eje horizontal. Con cada día que pasa se actualiza el gráfico y se marca las horas restantes al final de la jornada.

En la Figura 4, la línea azul indica la evolución real del proyecto y está formada con las horas restantes indicadas día a día. En este gráfico se puede ver que el equipo tuvo un desfase de diez horas el día seis y que fue recuperado para el día ocho. Esta figura, junto a la Figura 5 son generadas a partir de la información obtenida de un proyecto personal del alumno.

Si la curva de evolución real del desarrollo se aleja demasiado de la curva de tendencia a seguir, indica que el mecanismo de estimación tiene que ser ajustado. Una curva real por sobre la de tendencia mostraría un desplazamiento de la fecha de entrega lo que comúnmente se considera que el proyecto se encuentra atrasado y una curva real bajo la de tendencia mostraría que la fecha de entrega sería previa a la estimada.

2.8.2 Gráfico "*Burn up*"

Los gráficos "*Burn down*" tienen un gran problema, que se produce al basarse en las horas remanentes para terminar las tareas pendientes. Como sólo considera horas restantes, un proyecto que tenga todas sus tareas avanzadas, pero ninguna completada va a mostrarse con un estado normal y donde todo está funcionando de acuerdo al plan. En este desarrollo no se ha generado valor dado ya que ninguna funcionalidad o característica ha sido completada.

Los gráficos "*Burn up*" diagraman la cantidad de tareas, funcionalidades o historias de usuarios completadas y la cantidad a cumplir objetivo.

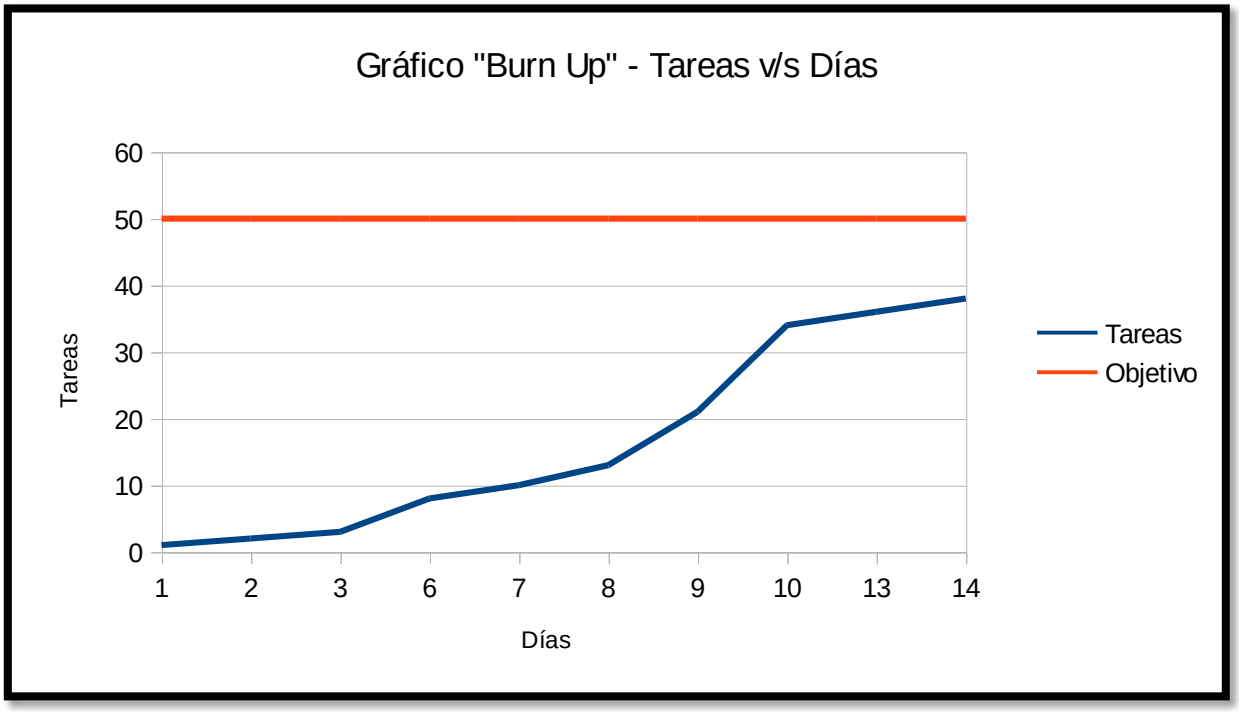

FIGURA 5 Gráfico "Burn Up"

Este gráfico (Figura 5) sí indica las tareas, funcionalidades o historias de usuario que han sido terminadas y que van de la mano al valor generado en el desarrollo. La tendencia es que este tipo de gráficos tenga una forma de "S" que dificulta el poder estimar una fecha de término a partir de la evolución del desarrollo atentando contra la predictibilidad buscada.

La Figura 5 es un ejemplo de gráficos *"Burn Up"* a mitad de un ciclo de cincuenta tareas. Se puede apreciar en el gráfico dos ciclos donde se generó conocimiento y luego éste se aplicó para generar las tareas siguientes. Estos ciclos corresponden a los periodos comprendidos entre los días uno y seis y entre los días seis y diez.

Al igual que en el gráfico "*Burn Down*", en el gráfico "*Burn Up*" se pueden identificar cuellos de botella a partir de la forma de la curva, pero no entregan mayor información al respecto. Para un equipo con baja experticia, la curva de la evolución del proyecto podría tener una forma que produzca confusión al interpretarse como algo normal. Nuevamente en la Figura 5, se ve que en el día diez algo ocurrió que hizo que la pendiente de la curva disminuyera por lo que se podría pensar que a partir de ese día el desarrollo fue menos productivo, aunque esto no fuese así.

2.8.3 Trabajo en Progreso

"Sólo lo que ha sido entregado genera valor". Bajo esta premisa toda tarea que esté en progreso no genera valor por lo que no debe ser considerada al momento de calcular el valor generado, pero sí es importante al momento de querer estimar plazos y esfuerzo restante o la salud del desarrollo.

David J. Anderson en su libro "*Agile Management for Software Engineering – Applying the Theory of Constraints for Business Results*" [1] considera el trabajo en progreso como el inventario de conocimiento, ideas para software de valor funcional. A estas ideas se le aplican distintas transformaciones hasta convertirse en el software que es entregado. Estas transformaciones podrían ser análisis, diseño, implementación, pruebas.

El proceso de diseño fue modelado por Marvin Patterson en el libro "*Accelerating Innovation*" [13] como un proceso de descubrimiento de información. A medida que el diseño emerge cada vez hay menos incerteza y más certeza hasta que el diseño esta completo. Poppendieck [14] toma este proceso de descubrimiento de información y plantea que todo proceso de desarrollo de software es un problema de diseño. Este planteamiento permitió modelar el flujo de valor como una reducción gradual de la incerteza y el descubrimiento de información detallada hasta que el código pasa las pruebas de control de calidad y es producido.

Ya pudiendo modelar el flujo de valor, Reinertsen introdujo dos nuevos conceptos en "*Managing the Design Factory*" [16]. El primero es que el inventario de ideas puede ser gráfico con diagramas de flujo acumulado y el segundo es que el valor del diseño deprecia en el tiempo.

Que el valor del diseño deprecie en el tiempo se produce por varias razones entre las cuales se destaca la variabilidad de los mercados: lo que puede ser muy interesante el día de hoy, ya no lo sea el día de mañana. También influyen situaciones como que la línea de suministro o de distribución se rompa, cambio en las leyes o regulaciones vigentes. Una idea que genera valor a partir de la diferenciación con la competencia lo hará mientras la competencia no copie la idea, cuando esto ocurra, ya no habrá diferencia y esta idea dejará de generar valor por diferenciación.

2.8.4 Ley de Little para desarrollo de productos

Un sistema de colas es un modelo donde los consumidores arriban y son agregados a una cola, esta cola es atendida por *n* servidores. Una vez que el consumidor es atendido, éste deja el sistema y el servidor atiende al siguiente consumidor.

Para un sistema de cola en estado de equilibrio, el largo promedio de la cola es equivalente al promedio de arribo multiplicado por el tiempo promedio de espera.

$$L_c = \lambda T_e$$

La Ley de Little puede ser transformada para adaptarse a un desarrollo de productos y escribirse del siguiente modo:

$$(Salida) = (Trabajo\ en\ progreso)/(Tasa\ promedio\ para\ completar\ una\ tarea)$$

De esto se desprende que el principal conductor del tiempo de ciclo, es el trabajo en progreso. Con esto, se muestra que el tamaño del trabajo en progreso es importante dado que en los desarrollos agiles se desea que el software sea entregado lo más seguido posible y en períodos cortos.

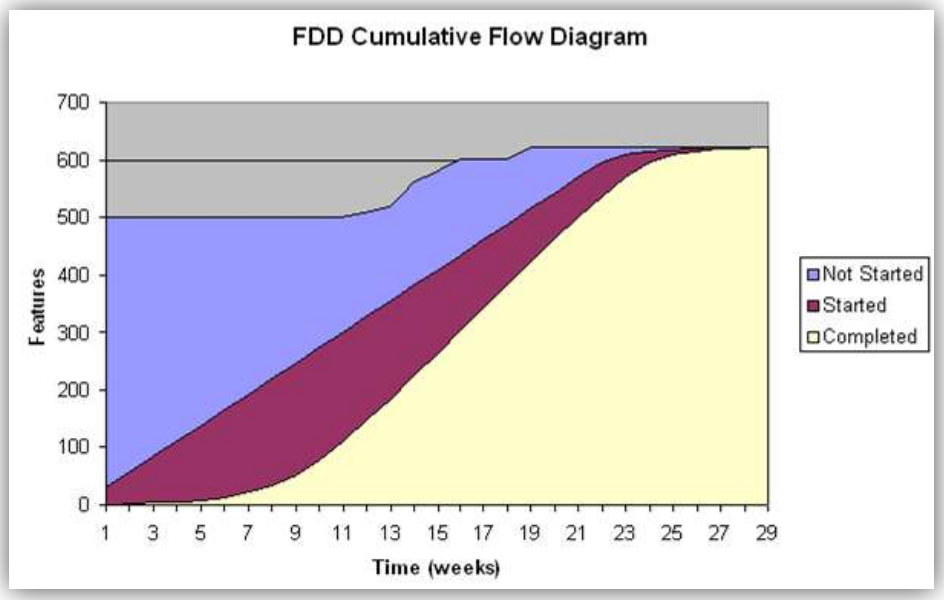

FIGURA 6. Ejemplo 1 De Diagrama De Flujo Acumulado [17]

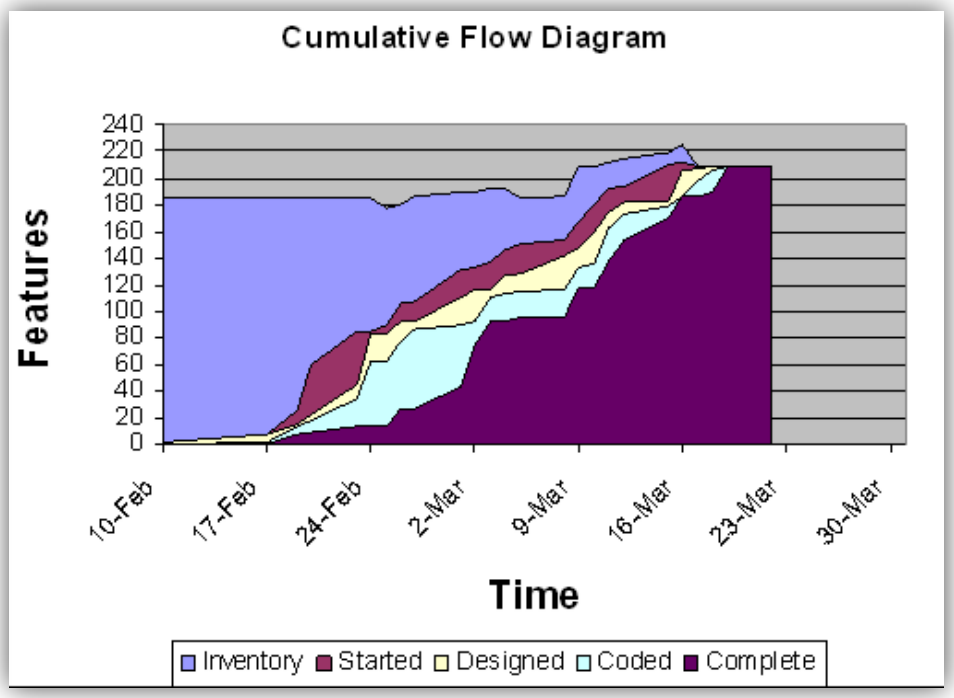

FIGURA 7. Ejemplo 2 de Diagrama de Flujo Acumulado [17]

La Figura 6 y Figura 7 son ejemplos de diagramas de flujo acumulado de iteraciones de alcance variable; esto quiere decir que los nuevos requerimientos pueden ingresar a los ítems por hacer. En el diagrama de la Figura 6 se muestra un desarrollo ideal donde la tasa de comienzo de tareas es constante durante la mayor parte del proyecto y las tareas completadas tiene la forma de una "S" característica de los proyectos donde se requiere generar conocimiento en un principio.

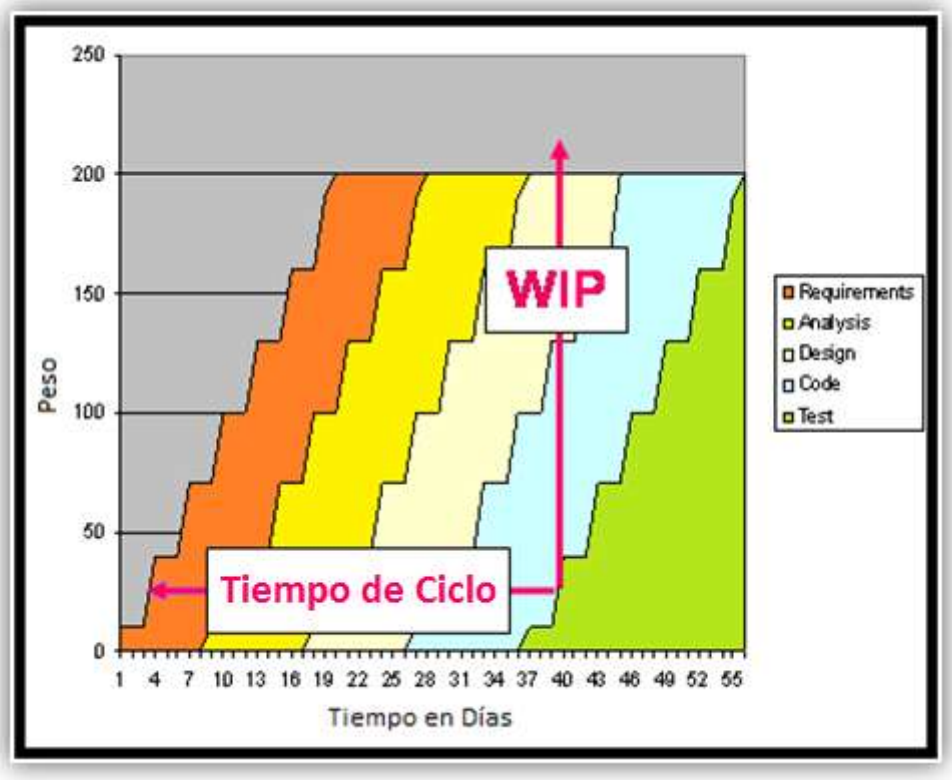

FIGURA 8. Representación de Diagrama de Flujo Acumulado [17]

La Figura 8 es una representación de la evolución de un proyecto cuyos requerimientos van siendo resueltos en grupos. Este gráfico sirve para ejemplificar como se interpreta el tiempo de ciclo en un diagrama de flujo acumulado y el trabajo en progreso (WIP).

Volviendo a la Figura 7, se puede apreciar el efecto de limitar el trabajo en progreso de la Figura 8 logrando mejorar los tiempos de ciclo y generando entregables los antes posible.

3 Desarrollo de la herramienta

3.1 Definición de estados

Todos los estados registrados en la aplicación pueden ser clasificados en tres grandes grupos: los que representan lo que se debe hacer en algún momento, lo que se debe hacer ahora y lo realizado.

Al primer grupo, que representa lo que alguna vez se debe realizar, le llamaremos "*Por Hacer"*. Esto hace analogía a las deudas pendientes que el equipo tiene con el cliente, pero que para el cliente no son importantes de pagar en el corto plazo.

Al segundo grupo, que representa la deuda del equipo en el corto plazo, le llamaremos *"En Curso"*. Este grupo incluye todos los estados por donde debe transitar una tarea antes de declararse terminada por parte del equipo de desarrollo Y al tercer grupo, que representa a los estados que indican que una tarea fue entregada para que sea puesta en producción se le llamara *"Hecho"*.

Los primeros estados que se ocuparon fueron **Por Hacer**, **Haciendo** y **Hecho**. Estos estados bastan para modelar un proceso de desarrollo básico donde el cliente ingresa los requerimientos en el estado Por Hacer, luego, el miembro del equipo que comience a trabajar en ellos, pasará la tarea al estado Haciendo y al terminar la dejará en estado Hecho.

Se hizo necesario agregar un estado que representara las tareas que el equipo dio por terminadas, pero que el cliente no ha validado. En un principio a este estado se le llamó "Validando", pero al convertirse éste en un posible cuello de botella, por lo lento que puede llegar a ser un paso a producción, se agregó otros estados llamados "Por Instalar" e "Instalado".

De este modo, una vez terminado el desarrollo de una tarea ésta debe ser validada por otro miembro del equipo realizando una prueba cruzada y pasada al estado Por Instalar. No se agregó un estado que corresponda a la validación por parte del cliente dado que éste es parte del desarrollo y está al tanto del producto liberado. De no tener al cliente durante la fase de

validación de la tarea se recomienda agregar un estado entre Validando y Por Instalar para contener las tareas pendientes de validación por parte del cliente.

Es importante tener en cuenta que la finalidad de las metodologías ágiles es generar valor lo antes posible y una de las maneras de lograr esto es poniendo en producción lo antes posible las funcionalidades desarrolladas. Es por esto que al pasar una tarea al estado Validando esta debería tener prioridad por sobre otras.

Para el caso en que el cliente no realice pasos a producción de forma periódica, sino sólo una al final del proyecto, entonces el estado Instalado pierde sentido y podría fusionarse con el estado Por Instalar. Cada tarea será instalada al final de las iteraciones por lo que el indicador de tiempo de ciclo resultaría constante para dicho cliente y no aportaría información para tomar decisiones.

3.2 Indicadores

Inicialmente se considero utilizar una serie de indicadores que incluían el número de cambios de estados y el número de retrocesos que una tarea tenía. A continuación se detallan los indicadores considerados, los no considerados y la razón de porque incluirlos o no. Los criterios que se utilizaron son la utilidad de los indicadores para el cliente del equipo y para el equipo.

a) Número de cambios de estado de una tarea:

   Al saber cuántos cambios de estado tiene una tarea, se puede identificar cuáles son los estados por donde más transitan éstas. Teniendo identificados los estados es posible analizar el porqué de esto y proporcionar una estrategia de mejora que permita agilizar el flujo de trabajo.

   Este indicador perdió sentido al comprobar que en la práctica las tareas avanzan en los estados y son pocas las veces que retroceden. En el caso de encontrar un defecto en una tarea, éste se agrega como una nueva tarea y queda asociada a la tarea que origino el fallo y una vez que todos los fallos hayan sido resueltos la tarea se da por terminada. Lo mismo ocurre si la nueva tarea es una mejora a la funcionalidad.

b) Número de retrocesos de una tarea:

El objetivo de este indicador es similar al anterior dado que busca saber qué estado es el que más rechaza tareas. Si hay un estado que rechaza muchas tareas puede significar que las políticas de aprobación son demasiado estrictas o que las políticas del estado anterior son demasiado blandas. En cualquiera de los dos casos este indicador desencadenaría una revisión de las políticas de aprobación de los estados del proceso. Al igual que en el indicador de número de cambios de estado, el indicador número de retrocesos fue descartado porque las tareas generalmente no retrocedían.

c) Estado con mayor cantidad de fallos reportados:

Este indicador fue introducido pensando en reemplazar b). Este indicador busca el estado en el cual se reportan mayor cantidad de fallos manteniendo el mismo objetivo que b).

Basado en la metodología propuesta, el estado "validando" tiene como funcionalidad comprobar la correctitud y completitud de las tareas por lo que este es el estado donde más fallos se encontrarán. En este escenario el indicador no entregaría información que permitiera mantener la mejora continua.

3.3 Requerimientos

3.3.1 Requerimientos no funcionales

Los requerimientos no funcionales de la aplicación se basan en mantener la simplicidad de la misma y su fácil utilización.

RN01. Simpleza para sincronizar el *Kanban* con el virtual. Es clave para el éxito de la herramienta que mantener el *Kanban* y el *Kanban* virtual sincronizados sea muy simple y sencillo para que la información entregada por la herramienta sea fidedigna.

RN02. Simpleza para recuperar el estado del *Kanban* a partir del *Kanban* virtual. Una de las funcionalidades del *Kanban* virtual es darle respaldo al *Kanban* en caso que este sufra de daños o pérdida de información, dado que el *Kanban* es la herramienta para poder gestionar los proyectos en los equipos que lo han adoptado.

3.3.2 Requerimientos Funcionales:

R01. El usuario debe poder ver la lista de tareas en el sistema con sus características. Las características de una tarea son todas aquellas propiedades que definen una tarea, como por ejemplo el estado y el nombre.

R02. El usuario debe ser capaz de ingresar nuevas tareas en la aplicación.

R03. El usuario debe ser capaz de cambiar una característica de una tarea a excepción del identificador único.

R04. La aplicación dejará un registro de los cambios de estado de una tarea. El registro de los cambios de estado de la tarea es necesario para el posterior cálculo de los indicadores de la misma.

R05. La aplicación debe poseer un mecanismo para configurar los estados del *Kanban*.

R06. La aplicación debe poder configurar los estados de inicio y fin para el cálculo de los indicadores.

R07. La aplicación debe entregar los indicadores por cada tarea.

R08. La aplicación debe generar el gráfico de flujo acumulado.

R09. El usuario debe poder rescatar la última configuración del *Kanban* conocida. Esto se requiere para poder recuperar el *Kanban* a partir del *Kanban* virtual en caso que este sufra de daños o perdida de tareas.

3.4 Versión mínima funcional

La meta de esta versión fue lograr registrar los eventos de cambio de estado de las tareas del mismo modo que lo hace una bitácora.

3.4.1 Modelo de datos

El modelo de datos consistió en tres columnas que indicaban el nombre de la tarea, la fecha del evento y el estado de destino (Figura 9).

	A	B	C
1	**Tarea**	**Estado**	**Fecha**
2	UI1	DONE	12/04/10 15:35
3	UI3	TODO	13/04/10 11:51
4	Ui1	WIP	12/04/10 14:55
5	M3	TODO	08/04/10 18:04
6	Ui1	TODO	12/04/10 12:54
7	UI1	Delivered	12/04/10 20:46

FIGURA 9. Modelo de datos original

3.4.2 Metodología de uso de la herramienta

Se utilizó como interfaz de usuario una planilla de cálculo Excel donde cada persona ingresaba un registro cuando realizaba un cambio de estado de una tarea.

El libro Excel que contenía la aplicación contaba con una hoja que hacía de bitácora y una hoja que tenía los indicadores por tarea.

3.4.3 Calcular indicadores sobre el modelo de datos

Se programó una macro que procesaba los datos ingresados en la bitácora para poder desplegarlo como indicador.

3.4.4 Retrospectiva

Surgió como un problema la falta de disciplina para registrar los cambios de estado cuando estos se realizaban. Los usuarios de la herramienta registraban los cambios de estado una vez a la semana lo que repercutía en imprecisión de los datos y la obtención de indicadores erróneos.

Cómo el objetivo de este trabajo es el poder respaldar el tablero *Kanban* físico y esto no se puede realizar si no se cuenta con datos confiables, entonces se requirió una nueva versión que abordase el problema.

3.5 Versión Web

El objetivo de esta versión es dar flexibilidad a los usuarios para que definan algunos parámetros para el cálculo de los indicadores y facilitar la sincronización.

3.5.1 Modelo de datos

El módulo de datos se convirtió a un modelo entidad relación con tres tablas: los estados, la bitácora y la configuración. La evolución y la capacidad de la herramienta de adaptarse a las decisiones del equipo que la usa, son clave, por agregar la posibilidad de configurar cuales son los estados de inicio y fin.

3.5.2 Metodología de uso de la herramienta

Al momento de registrar un cambio de estado de la tarea el usuario ingresa al sistema que le provee de un cuadro de texto y un campo de selección donde ingresa el nombre de la tarea y el estado de destino. Se decidió no incluir el campo para ingresar la fecha del evento para impedir que el usuario pudiera registrar los cambios con posterioridad.

FIGURA 10. Ventana Tareas Versión Web

FIGURA 11. Detalle de tareas versión web

La interfaz de la aplicación mostrada en la Figura 10, muestra la lista de tareas registradas en el sistema agrupadas por el estado en el que se encuentran. Una vez presionada la flecha que se encuentra a un costado del nombre de la tarea se puede acceder al detalle de esta como se muestra en la Figura 11.

FIGURA 12. Registro de cambios de tareas versión web

Al presionar el "Registrar Cambio" de la Figura 11 se llega a ventana mostrada en la Figura 12. En esta ventana se selecciona el nuevo estado de la tarea y para guardar el cambio se presiona "Agregar" dejando agregado en el sistema este nuevo cambio.

3.5.3 Calcular indicadores sobre el sistema

Se migró el cálculo de los indicadores de la macros de Excel al lenguaje de la aplicación web cambiando el origen de los datos.

3.5.4 Retrospectiva

Los objetivos de esta versión se cumplieron al aumentar la cantidad de datos confiables y que se convirtiera en parte de la cultura el registrar los cambios de las tareas en la herramienta junto con el hacerlo en el *Kanban*.

3.6 Versión 3.0

3.6.1 Metodología

Después de analizar las demás alternativas de solución, se llegó a la conclusión de que para desarrollar la alternativa más sencilla que cumpliera con los objetivos de la aplicación, ésta debía cumplir con las facilidades de manipulación del *Kanban* físico, por lo cual se agregaron botones de fácil acceso para las acciones más recurrentes como lo son el crear y editar tareas.

Una vez puesta en producción la primera versión web se concluyó que ésta era muy lenta. Por esto se agregó un nuevo objetivo que consistía en lograr una interfaz gráfica que fuese rápida y no limitara las acciones simultáneas que se podían realizar.

La siguiente versión de la aplicación se pensó como una interfaz enriquecida que usase llamadas asíncronas para interactuar con el servidor. Para esto se decidió utilizar la librería *javascript extjs* versión 3.3.1 para implementar la nueva interfaz gráfica. El modelo de datos y la estructura general de ésta versión es la misma que la de la versión anterior.

nombre	Estado	peso
Setear estado por defecto al crear tarea	Por Hacer	1
Crear servicio para cambiar estado de una tarea	Hecho	1
Setear icono de agregar tarea	Hecho	1
Crear servicio para registrar tiempo consumido	Hecho	2
Crear ventana para registrar tiempo consumido	Hecho	1
Crear servicio para eliminar tarea	Por Hacer	1
Registrar tareas pendientes	Haciendo	5
Setear icono de borrar una tarea	Hecho	1
Generar calculo de indicadores	Hecho	3
Crear ventana para configurar estados	Haciendo	6
Aplicar filtros de estado	Hecho	1
Levantar aplicación en servidor	Hecho	4
Aplicar filtro por fechas	Hecho	1
[BUG] El filtro por fecha no incluye las tareas actualizadas durante el día seleccion	Hecho	0
Crear descripción de uso de aplicación	Haciendo	2
Registrar los cambios de estados de las tareas	Hecho	1
Encontrar iconos agregar y borrar tarea armonicos	Hecho	1
[BUG] Actualizar tiempo consumido en el detalle	Por Hacer	0
Modificar servicio de estados para filtrar los habilitados	Por Hacer	1
Agregar propiedad habilitado a los estados	Hecho	1
[BUG] No se puede borrar lo seleccionado en el filtro de fechas	Por Hacer	0

FIGURA 13. Vista De Versión 3.0

La vista (Figura 13), con forma de grilla, permite que el usuario ordene las columnas de manera alfabética para facilitar la búsqueda de una tarea y que los usuarios tuvieran toda la información necesaria a simple vista para facilitar la sincronía con el *Kanban* físico. Siguiendo el mismo objetivo, la funcionalidad de agregar una tarea consiste en registrar en la ventana para agregar tareas el estado en el que ingresa al *Kanban* y el peso correspondiente.

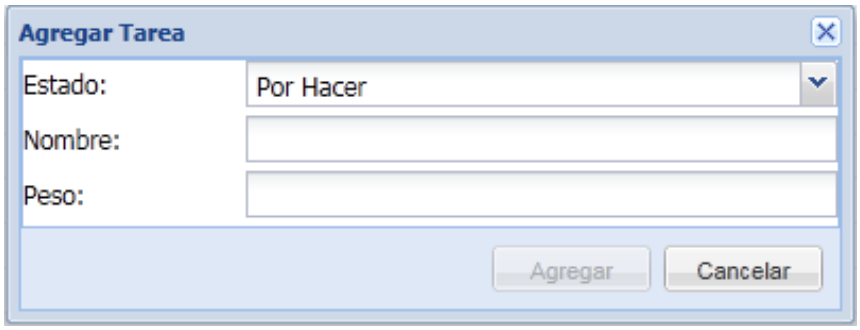

FIGURA 14. Agregar Tarea Versión 3.0

Cómo la sincronización tiene que ser algo simple y de fácil acceso, la edición de una tarea se realiza en una barra lateral donde se puede cambiar el nombre, el estado y el peso de la tarea como se muestra en la Figura 14.

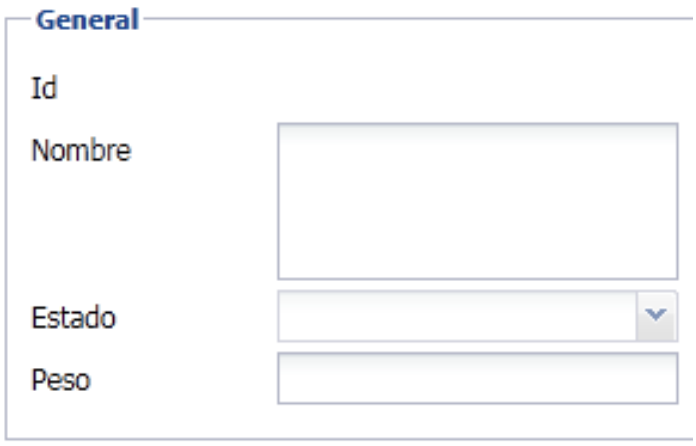

FIGURA 15. Edición De Tarea Versión 3.0

Para activar la edición se debe seleccionar la tarea a editar y la información de esta se cargará en la barra con el detalle. La Figura 15 muestra la interfaz de edición de una tarea sin datos.

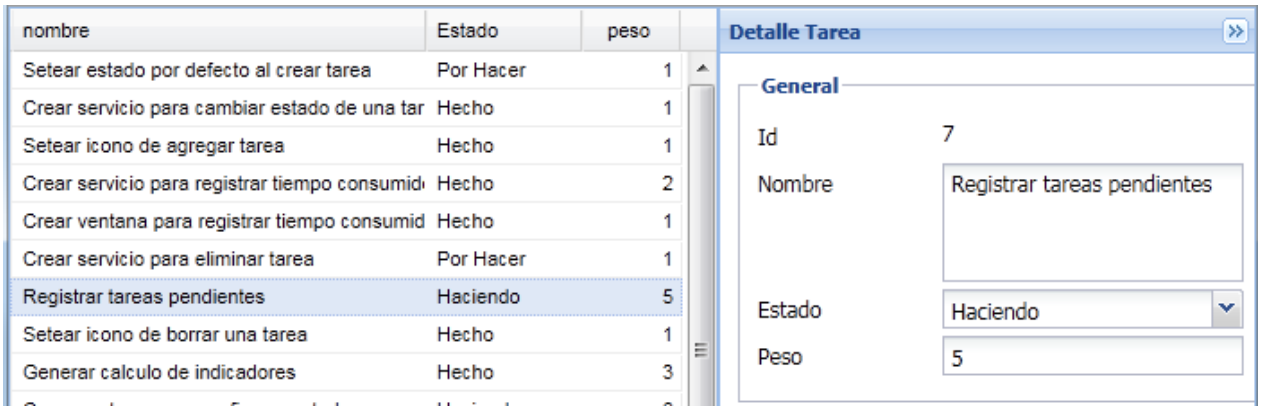

FIGURA 16. Detalle de Tarea Versión 3.0

La Figura 16 muestra una tarea seleccionada y el detalle de la misma en la ventana de edición.

3.6.2 Retrospectiva

Esta versión de la aplicación da soporte limitado al *Kanban* dado que no permitía agregarle nuevas dimensiones a las tareas, sólo poseía nombre, estado y peso. Los *Kanban* pueden tener tantas dimensiones como el equipo que los utiliza requiere y esta herramienta no debe ser una limitante para esto.

Junto con lo anterior, la aplicación con la información actual no es capaz de calcular tiempo de ciclo, de ingeniería ni la relación peso/tiempo.

3.7 Versión 4.0

La versión 4.0 debe suplir con las falencias presentadas por la versión 3.0 representadas por los siguientes requerimientos.

1. Poder saber la relación peso/esfuerzo promedio de las tareas, se requiere:
    - Conocer el estado que representa el termino del desarrollo de una tarea.
    - Una funcionalidad que permita registrar la cantidad de horas trabajadas en una tarea.
    - Agregar a la grilla las horas consumidas.
    - Motor de cálculo de la relación.

2. El segundo requerimiento consiste en poder calcular el tiempo de ciclo y tiempo de ingeniería de una tarea específica. Se requiere:
    - Conocer los estados de inicio y término del tiempo de ciclo y del tiempo de ingeniería.
    - Conocer cuando se realizaron los cambios de estado de una tarea.
    - Motor de cálculo de cada indicador.
3. Permitir que una tarea posea n dimensiones, a estas dimensiones le llamaremos propiedades de una tarea. Se requiere:
    - Agregar lista de registros con forma llave valor a las tareas.
    - Agregar a la grilla las propiedades.
    - Proveer vista de edición de propiedades.

Las maquetas diseñadas para implementar estas tres funcionalidades son las siguientes tres imágenes.

nombre	Estado	peso	Propiedades		tiempo	Actualizado
Setear estado por defecto al crear tar…	Por Hacer	1	**tipo**	Tarea	0.00	23/06/2011 17:1…
Crear servicio para cambiar estado d…	Instalado	1	**proyecto**		0.00	23/06/2011 17:1…
			prioridad	Base 2		
Setear icono de agregar tarea	Instalado	1			0.00	28/05/2011 13:2…
Crear servicio para registrar tiempo c…	Instalado	2			2.00	29/05/2011 19:3…
Crear ventana para registrar tiempo c…	Instalado	1			3.00	29/05/2011 19:3…
Crear servicio para eliminar tarea	Por Hacer	1			0.00	17/05/2011 23:4…
Registrar tareas pendientes	Haciendo	5			0.00	20/05/2011 02:0…
Setear icono de borrar una tarea	Instalado	1			0.00	28/05/2011 13:2…
Generar calculo de indicadores	Instalado	3			3.00	11/06/2011 23:5…
Crear ventana para configurar estados	Haciendo	6			8.00	06/06/2011 23:2…

FIGURA 17. Grilla de Tareas Versión 4.0

La maqueta presentada en la Figura 17 muestra la interfaz diseñada para satisfacer parte del requerimiento primero y tercero. Esta interfaz permite ver el tiempo consumido por una tarea y las propiedades de ésta.

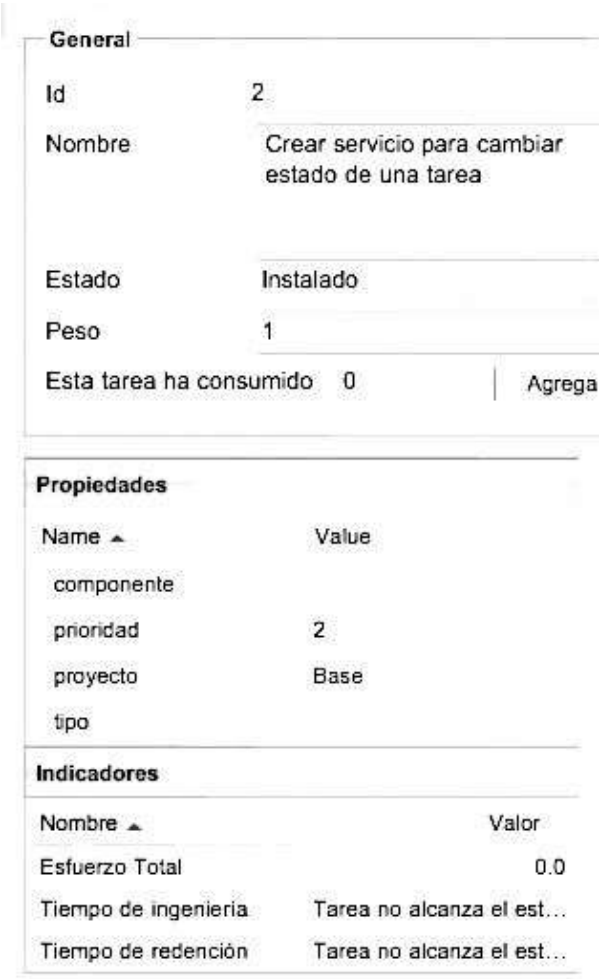

FIGURA 18. Detalle Tareas Versión 4.0

La Figura 18 completa los requerimientos primero y tercero al proveer la funcionalidad de agregar horas consumidas y editar las propiedades de una tarea.

FIGURA 19. Barra de Filtros Versión 4.0

Para facilitar la sincronización entre el Kanban y el Kanban virtual, la aplicación cuenta con una serie de filtros descritos previamente como se muestra en la Figura 19.

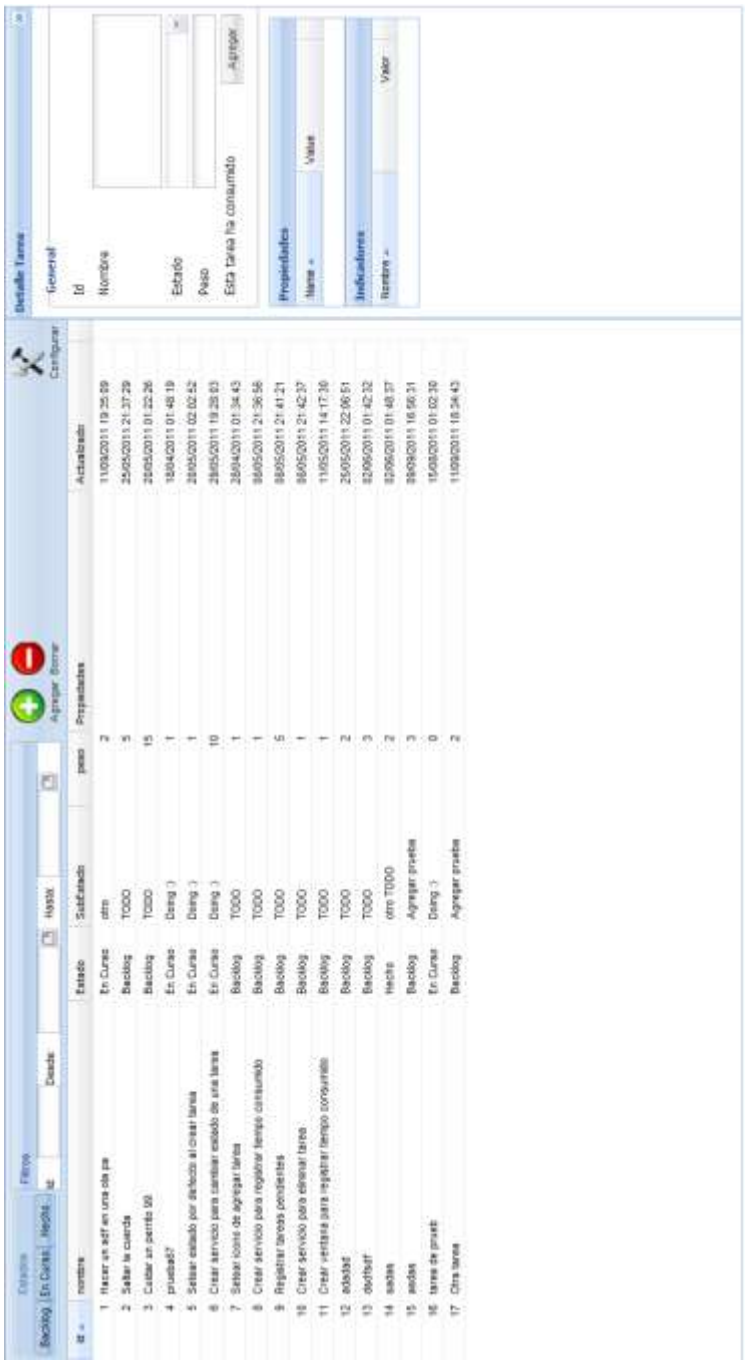

FIGURA 20. Interfaz Aplicación Final - Pantalla Principal

La interfaz presentada por la Figura 20 muestra la visión general de la aplicación. A la barra de filtros se le agregó la posibilidad de buscar por el id de la tarea con el fin de facilitar la búsqueda de estas y poder acceder a la vista de detalle.

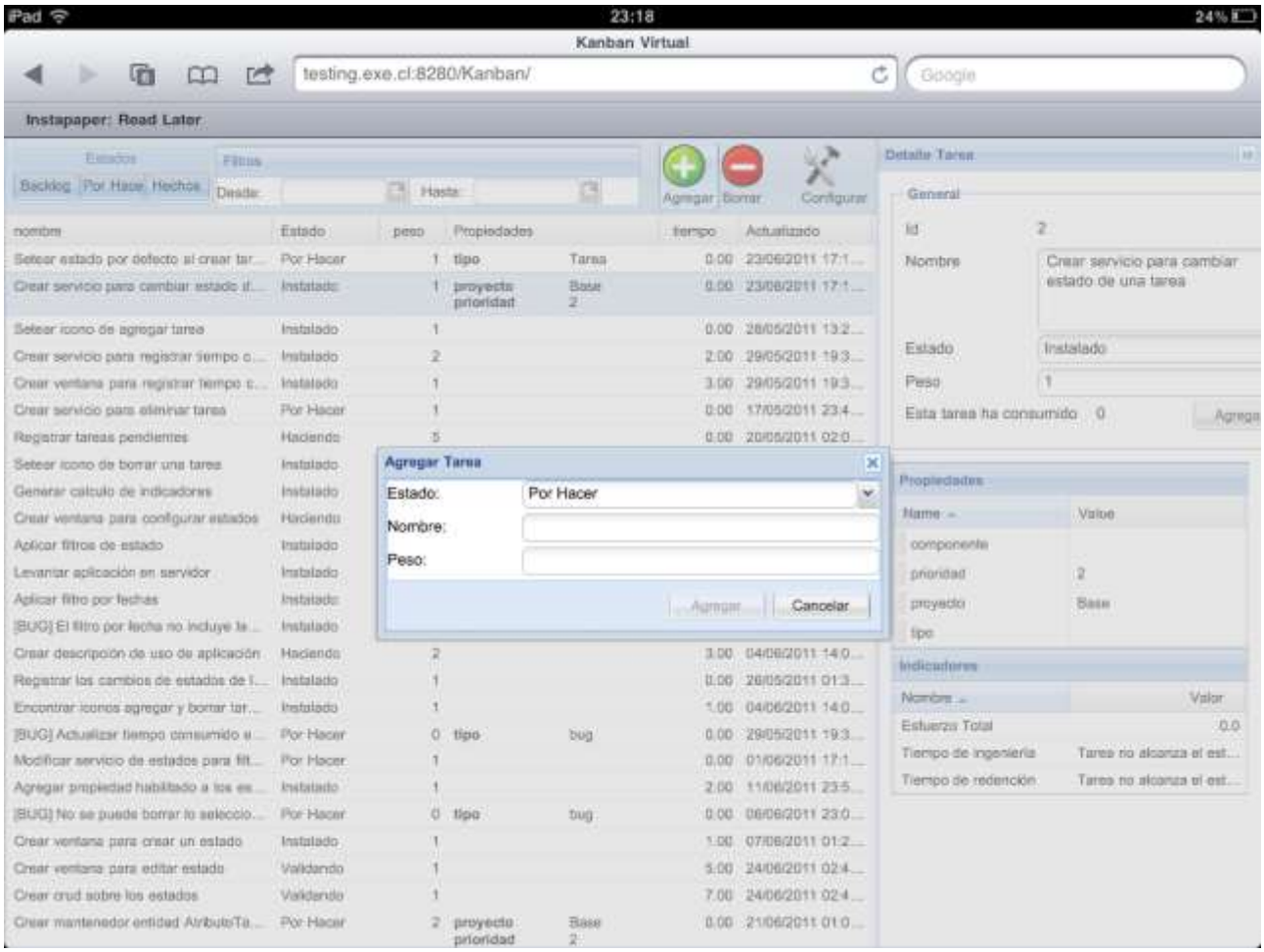

FIGURA 21. Interfaz Aplicación Final - Agregar Tarea

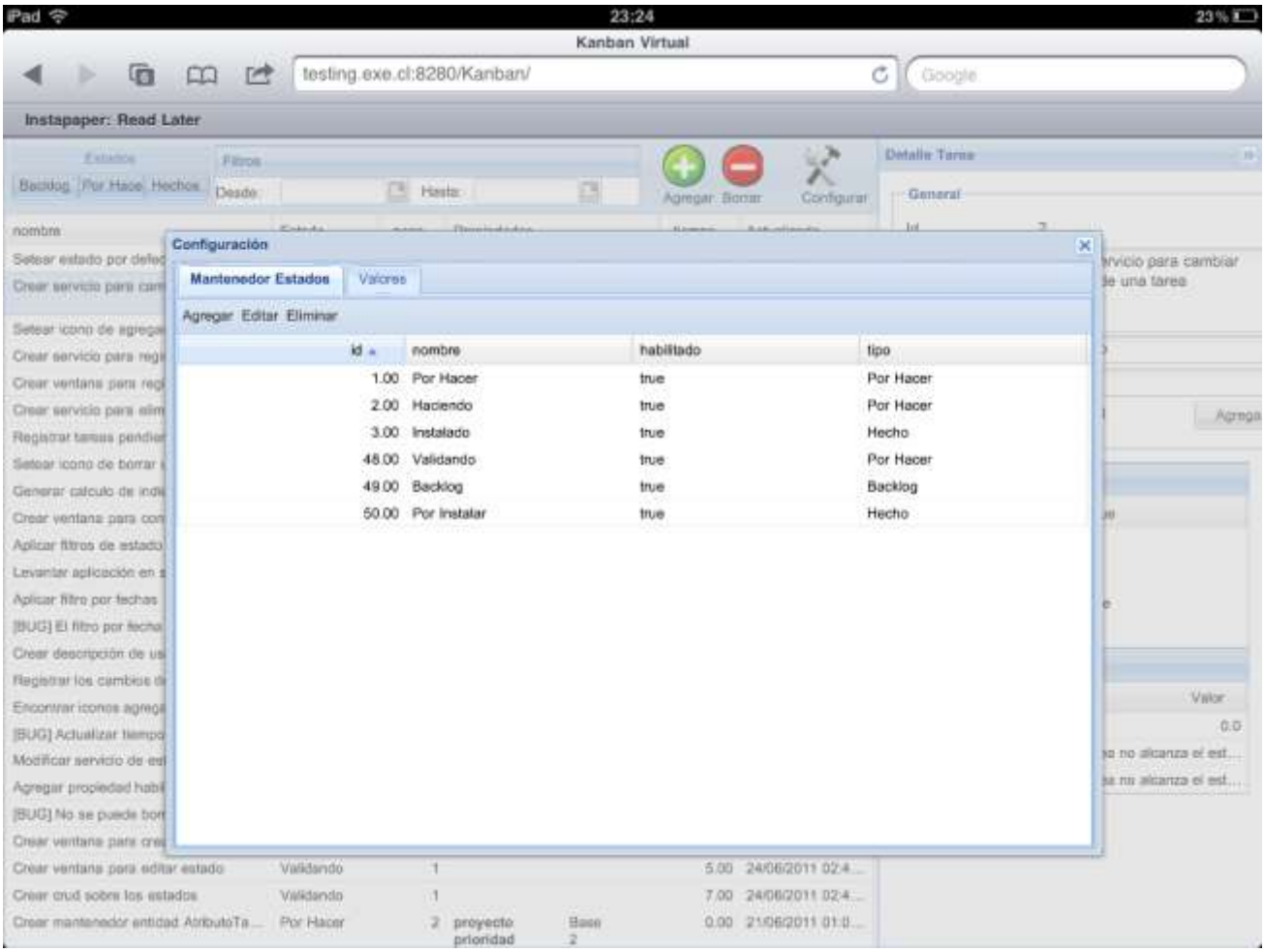

FIGURA 22. Interfaz Aplicación Final – Configuración

La Figura 21 muestra como agregar una nueva tarea en la versión final y la Figura 22 muestra la configuración de los estados de la aplicación.

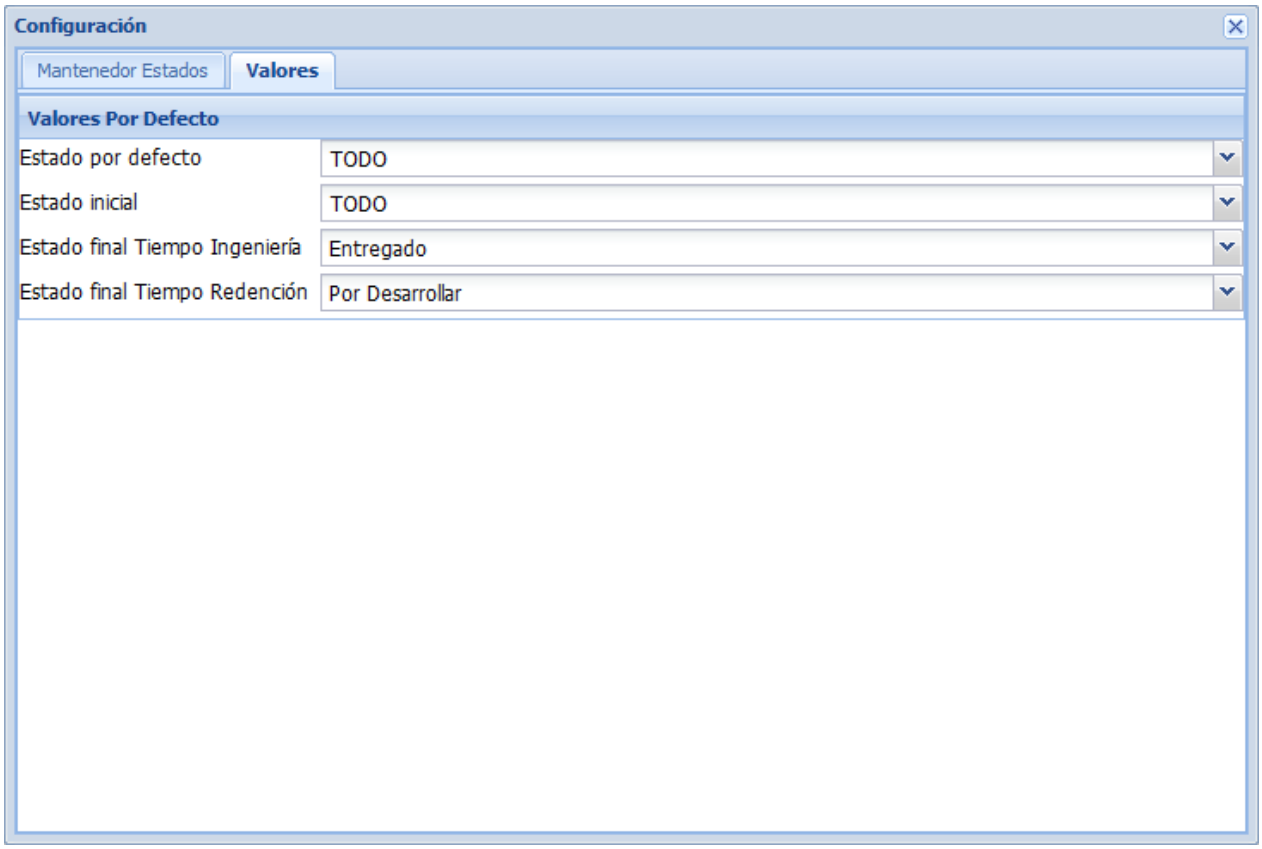

FIGURA 23 Interfaz Aplicación Final – Configuración General

La interfaz de la Figura 23 es el menú de configuración donde se puede cambiar el estado por defecto que la aplicación muestra al momento de agregar una nueva tarea. La importancia de esta ventana radica en que en ella se pueden configurar el estado inicial y los estados finales del tiempo de ingeniería y redención dándole al equipo flexibilidad.

3.8 Creación de Reporte de Flujo Acumulado

Dada la importancia de los diagramas de flujo acumulado para el manejo y control de la salud de los desarrollos ágiles, el *Kanban* virtual también debe calcularlo.

La aplicación registra cada cambio de estado de una señal en el tablero guardando la fecha, el estado anterior y el nuevo estado. Con esta información *Kanban* virtual es capaz de generar el diagrama de flujo acumulado entre dos fechas.

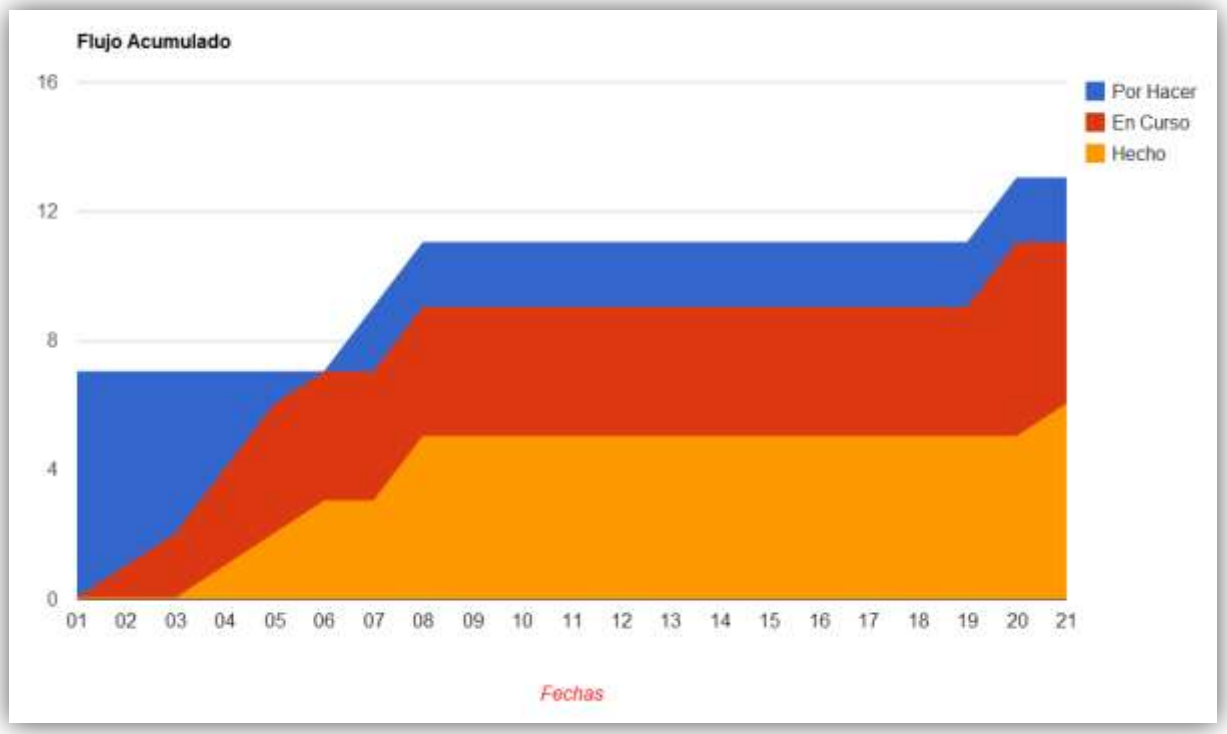

FIGURA 24. Diagrama de Flujo Acumulado

La Figura 24 muestra el flujo acumulado entre el día 01 y el día 21 del desarrollo de este Trabajo de Título. Entre los días 11 y 13, dos tareas pasaron a estar hechas, pero otras dos fueron devueltas a en curso al descubrírsele desperfectos, lo que para efectos prácticos no produjo cambios en el flujo acumulado.

3.9 Evolución de la herramienta

El desarrollo de la herramienta fue variando en base a las necesidades de la misma y siempre enfocado a mantener la flexibilidad que ofrece el tablero *Kanban*.

Como primera aproximación, se realizó un modelo sencillo y básico en Excel, para realizar pruebas de funcionalidad. Una vez probado se actualizó traspasándose a Java para poder realizar las primeras mejoras.

El segundo paso fue mejorar la interfaz gráfica, haciéndola más amigable y fácil de utilizar por cualquier usuario que utilice un *Kanban*.

Una vez que se logró obtener una herramienta amigable, se procedió a optimizarla. Se integraron los cálculos de indicadores, implementación de mejoras como los diagramas de flujo acumulado, y el trabajo final se enfocó en mejorar las falencias del *Kanban*, o sea se siguió hacia el cumplimiento del objetivo principal de este Trabajo de Título.

Se logró obtener una herramienta que puede ser utilizada sin problemas por cualquier usuario que trabaje con *Kanban*, amigable y que contrarresta la mayoría de las falencias que presenta el *Kanban*. Es una herramienta que puede ser utilizada vía web, no sólo a través de computadores sino también de dispositivos móviles de tipo celulares y tablets, entregando mayor flexibilidad para su uso.

4 Experiencia de Uso

4.1 Empresas Privadas

4.1.1 Empresa de Desarrollo

Esta empresa está pasando por un proceso de adopción de las metodologías ágiles y anteriormente su metodología estaba basada en RUP. Muchos de los indicadores que la empresa utilizaba antes de comenzar con la adopción de la agilidad dependen de la metodología RUP por lo que se requiere registrar información de la tarea tal como la etapa en la cual se desarrolla.

En metodologías basadas en cascada existen etapas estrictas que deben terminarse completamente antes de comenzar la etapa siguiente, estas etapas podrían ser análisis, diseño, implementación e implantación. Cada tarea ingresada al *Kanban* pasaría por las mismas etapas, en un principio tiene que ser analizada y diseñada para ser implementada e implantada. Se pudo continuar considerando las mismas etapas al asumir que el tiempo necesario para los *Planning Game* y el de discusión con el cliente era parte de la etapa de análisis, las discusiones internas del equipo eran parte de diseño y la construcción parte de la implementación, pero con la gran diferencia de que las etapas no eran excluyentes y una tarea podía estar pasando por varias etapas en un mismo instante. Por ejemplo, en medio de la construcción podía ocurrir una reunión con el cliente para aclarar o cambiar el alcance de una tarea o realizar un nuevo *Planning Game* que cambie su prioridad, alcance y definición.

La empresa cuenta con dos líneas de producción, una que satisface los desarrollos específicos para clientes y otra que satisface las necesidades de los productos que ofrece bajo licencia. En el Tablero *Kanban* las tareas tienen las siguientes dimensiones:

- Proyecto al cual pertenece la tarea.
- Tipo de Incidencia. Si es una tarea normal o planificada, una falla o defecto o una mejora o nueva funcionalidad. Estos diferentes tipos tiene como funcionalidad facilitar la facturación de los proyectos.
- Prioridad que indica que tan importante o que tanto valor genera para el cliente.

- Fecha de Inicio.
- Fecha de entrega.
- Componente. Sirve para agrupar las tareas por módulo o funcionalidad. Por ejemplo, el componente de seguridad o de administración de usuario.
- Versión. Indica cuando una tarea va a ser liberada al cliente.
- Asignado o responsable de realizar una tarea.
- Tiempo original estimado.
- Etapa.
- Resumen.
- Unidad de Trabajo.
- Descripción.

Esta empresa cuenta con algunas prácticas de gestión ágil de proyectos como el *Stand up Meeting* y *Planning Game*. La funcionalidad de filtrar las tareas según su última modificación fue algo que, en un periodo de tiempo, funcionó bastante bien al dejar a simple vista las tareas que habían sido desarrolladas la jornada anterior, pero surgió la necesidad de incluir nuevos filtros.

De esta evaluación nace la necesidad de poder filtrar las tareas según alguna de las propiedades que éstas tengan registradas. Por ejemplo, les interesaba ver las tareas que pertenecían a un cierto proyecto o las que estaban asignadas a una u otra persona.

La Figura 25 muestra el Kanban utilizado por esta empresa donde las divisiones verticales corresponden a los estados por los que debe pasar cada tarea y las divisiones verticales los distintos proyectos. El tamaño de la columna es el límite de tareas que pueden encontrarse en ese estado.

FIGURA 25. Kanban utilizado en la empresa de desarrollo

Otra necesidad planteada por la empresa consistía en saber el tiempo consumido por un proyecto. Para esto la aplicación debería tener la funcionalidad de sumar el tiempo consumido por todas las tareas que cumplan con un criterio, lo que corresponde a aplicar el filtro por propiedades y luego calcular el tiempo consumido.

A partir del uso de la herramienta la empresa fue capaz de darse cuenta que el gran parte del tiempo de ciclo de una tarea se gastaba en el tiempo de redención y no en el tiempo de ingeniería como se creía. Junto con lo anterior, se cambió el límite de tareas por estado hasta llegar a los tiempos de ciclo deseados.

4.1.2 Empresa consultora área minera

Este caso fue particularmente distinto, ya que se trata de una empresa muy pequeña (cercana a los 10 trabajadores), casi todos de más de 50 años de edad, sin tener idea de que se trata la agilidad, muy acostumbrados a trabajar en distintos proyectos al mismo tiempo, muchas veces dependiendo de la información que le provea el cliente para poder avanzar. El rubro son estudios de mantenibilidad y confiabilidad de plantas mineras, trabajando para grandes empresas multinacionales de ingeniería y empresas mineras.

La facilidad estaba en que al ser pocas personas, es más fácil la implementación de un *Kanban*, pero la dificultad era que al ser personas acostumbradas toda su vida a trabajar de cierta manera, era más difícil que se acordaran de actualizar el *Kanban* y más aún la aplicación de respaldo. Al ser la primera vez que se aplica agilidad a esta empresa, la cultura necesaria para trabajar con el *Kanban* y actualizarlo, no existe y hay que partir de cero.

Al cabo de un mes, la mayoría de los trabajadores ya hacían uso activamente del *Kanban*, excepto por los de mayor edad que aún eran reacios a la utilidad que les podría traer un tablero como ese. Pero aún olvidaban realizar la actualización en la aplicación.

En el último reporte recibido de la empresa, la actualización de la aplicación se la habían encargado al secretario para que la hiciera al final de cada día de trabajo, para poder comenzar a contabilizar las horas que cada profesional utilizaba para cada proyecto en el que participaba. El *Kanban* ya se utilizaba casi al 100% en su versión más básica.

En esta empresa el tiempo de redención de las tareas era muy superior al tiempo de ingeniería, esto era producto de que las tareas eran abordadas lo más tarde posible dada la alta tasa de cambios que los clientes solicitaban. De esta manera, la empresa sólo le interesaba monitorear los tiempos de ingeniería caracterizando los tipos de tareas para mejorar la estimación.

4.1.3 Empresa Estudio de Diseño

Esta empresa se caracteriza por estar compuesta de 7 personas, sólo gente joven, de mente muy abierta, acostumbrados a usar tecnología, agilidad y *Kanban* para manejar los proyectos que llevan a cabo.

Ellos aceptaron usar la aplicación y desde un principio ayudaban reportando fallos y haciendo sugerencias de usabilidad y diseño. La utilizaban tanto en iPhones e iPads, haciéndola una herramienta de uso diario en la empresa.

La aplicación fue considerada bastante útil, sobretodo porque pudieron calcular indicadores nuevos que antes no habían podido calcular. Al igual que en la consultora de minería el foco fue el tiempo de ingeniería por la alta tasa de cambio de las tareas.

4.1.4 Uso en este Trabajo de Título

Como soporte y ayuda a la gestión del desarrollo de la aplicación y luego del informe se utilizó un *Kanban*.

Para este proyecto se requería que las tareas tuvieran las siguientes dimensiones:

- Prioridad, que indica la urgencia de la tarea
- Tipo, para indicar si es una tarea planificada, un defecto o una tarea relacionada con el informe

Las tareas eran ingresadas al *Kanban* y se les asignaba un peso inicial de cero hasta que se planificaba una nueva entrega donde a las tareas con mayor prioridad se les asignaba peso. Al ser entregas de plazo fijo se consideraban las tareas cuyo peso total no excediera los veinticinco puntos.

Esta forma de trabajo resultó ser efectiva al ser un proyecto donde todos los roles eran representados por una persona.

Dadas las condiciones de este proyecto y del uso que se le da al *Kanban*, los indicadores entregados por la aplicación no fueron de utilidad, pero fue crucial la información que la herramienta proveía para obtener la velocidad y poder planificar que tareas pertenecerían a la nueva liberación.

4.2 Mejoras propuestas

Una de las bases de la agilidad es entregar valor lo antes posible y eliminar el desperdicio por lo que generalmente los desarrollos se preocupan primero por hacer algo que solucione el problema y luego mejorarlo.

4.2.1 Sincronización de *Kanban* y *Kanban* virtual

Uno de los puntos que pueden mejorarse de la aplicación desarrollada es el no tener que actualizar el *Kanban* y el *Kanban* virtual por separado.

- Códigos QR

  Una de las maneras de hacer esto es utilizando códigos QR para poder reconocer las tareas mediante un sistema de reconocimiento de imágenes que permita obtener a partir de una configuración los estados en los que se encuentran las tareas. Este método tiene el problema de tomar una imagen y chequear el estado de cada una de las tareas para poder registrar un cambio de estado en la aplicación, lo que puede llevar a imprecisiones si es que estas fotografías son tomadas en intervalos muy extensos.

  Este método sería útil como medio de validación si lo que contiene el *Kanban* corresponde a lo que contiene el *Kanban* virtual.

- Tablero Táctil

  En la actualidad un tablero táctil es demasiado caro para gran parte de las organizaciones que adoptan agilidad, pero en un futuro el costo de estos tableros podría ser accesible. En este caso la aplicación podría integrarse con una nueva interfaz diseñada para emular el *Kanban* físico sacrificando la flexibilidad que un *Kanban* analógico brinda, pero ganando en la facilidad de sincronización y ser distribuido.

4.2.2 Incorporar en la interfaz de usuario elementos de gestión visual

Se propone el agregar algunos elementos de gestión visual de información al *Kanban* virtual con el fin de ayudar a la gestión de la información.

FIGURA 26. Gestión visual del Kanban físico

La Figura 26 muestra un *Kanban* donde los distintos tipos de tareas son marcados con distintos colores, del mismo modo, se podrían usar distintos tamaños o tipos de señales para indicar otro tipo de información. Al traspasar parte de esta información al *Kanban* virtual se facilitaría la tarea de mantener ambas herramientas sincronizadas.

5 Discusión y conclusiones

Usar soluciones de baja tecnología para resolver problemas en la producción de alta tecnología es algo que ha demostrado funcionar en una variedad de industrias alrededor del mundo. El *Kanban* es una herramienta de baja tecnología que ha dado frutos en áreas de alta tecnología como lo son la construcción de vehículos o el desarrollo de software gracias a su simplicidad y su capacidad de irradiar información, pero el hecho de ser de baja tecnología no implica que no pueda ser potenciado con herramientas de alta tecnología. La aplicación desarrollada es una herramienta de tecnología media por ser un software sencillo y poseer cálculos automáticos de métricas, que unida a una de tecnología baja como lo es el *Kanban* es capaz de dar soporte a gran parte de los procesos productivos.

Los objetivos de esta memoria lograron ser satisfechos al obtener una aplicación con la capacidad de recuperar el último estado de una tarea y poder recuperar la última configuración del tablero *Kanban*, lo que incluye los distintos estados de éste y las distintas dimensiones de las tareas con sus valores. Además, la aplicación es capaz de entregar indicadores para cada una de las tareas que cumplen con las reglas de cálculo de cada indicador y un historial de los cambios de estado de una tarea.

El modelo de datos sobre el cual se desarrolló la aplicación posee información con respecto a cuándo se realizan los cambios de estado de una tarea, el último valor de sus propiedades y tiempo total consumido. Con esta información se pueden generar reportes tales como gráficos de evolución de las tareas terminadas o calculo de indicadores de tareas agrupadas.

La flexibilidad del *Kanban* es uno de sus puntos fuertes y por tanto la aplicación debía serlo también. Esto se vio cumplido en los casos de prueba, dado que en los diferentes escenarios no hubo problemas para registrar las distintas configuraciones que los equipos habían definido para sus tableros y de la información que este proveía para su propia gestión.

Kanban como herramienta es simple de usar, pero puede adaptarse a los más complejos escenarios dejando al equipo el definir la interpretación de la sintaxis y la semántica que estos

nos entregan y es por esto que la aplicación sólo restringe que las tareas tengan estados, peso y nombre.

Bibliografía

1. **Riquelme, Roberto.** *Una herramienta expresiva para implementación de tableros Kanban virtuales.* s.l. : Universidad de Chile, 2011.

2. **Sarah Dámaris Amaro Calderón, Jorge Carlos Valverde Rebaza.** *Metodologías Ágiles.* s.l. : Universidad Nacional de Trujillo, Perú, 2007.

3. **Anderson, David J.** *Kanban.* s.l. : Blue Hole Press, 2010.

4. **Allue, Xavier Quesada.** Visual Management Blog. *Build a taskboard in 10 steps.* [En línea] 15 de Diciembre de 2009. http://www.xqa.com.ar/visualmanagement/2009/12/build-a-taskboard-in-10-steps/.

5. **Henrik Kniberg, Mattias Skarin.** *Kanban and Scrum, making the most of both.* s.l. : lulu.com, 2010.

6. **Ladas, Corey.** *Scrumban - Essays on Kanban Systems for Lean Software Development.* s.l. : Modus Cooperandi Press, 2009.

7. **Anderson, David J.** David J Anderson & Associates Management Consulting for Knowledge Workers. *Thoughts on how Kanban differs from scrum.* [En línea] 10 de Junio de 2010. http://agilemanagement.net/index.php/Blog/thoughts_on_how_kanban_differs_from_scrum/.

8. **Hiranabe, Kenji.** InfoQueue. *Kanban Applied to Software Development: from Agile to Lean.* [En línea] 14 de Enero de 2008. http://www.infoq.com/articles/hiranabe-lean-agile-kanban.

9. **Kniberg, Henrik.** *Lean from the Trenches.* 2011.

10. **Poppendieck, Mary and Tom.** *Implementing Lean Software Development: From Concept to Cash.* s.l. : Addison-Wesley Professional, 2006.

11. **Anderson, David J.** *Agile Management for Software Engineering - Applying the Theory of Constraints for Business Results.* Upper Saddle River NJ : Prentice Hall, 2003.

12. **Corbett, Thomas.** *Throughput Accounting.* Great Barrington MA : North River Press, 1997.

13. **Patterson, Marvin.** *Accelerating Innovation.* New York NY : Van Nostrand Reinhold, 1993.

14. **Poppendieck, Mary and Tom.** *Lean Software Development - an agile toolkit.* New York NY : Addison Wesley, 2003.

15. **Reinertsen, Donald G.** *Managing the Design Factory - A Product Developer's Toolkit.* New York NY : Free Press, 1997.

16. **Company, Toyota Global.** Just-in-Time — Philosophy of complete elimination of waste. [En línea] Toyota. [Citado el: 24 de September de 2011.] http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html.

17. *Using Cumulative Flow Diagrams.* **Anderson, David J.** s.l. : Embarcadero Technologies, Inc., 2004.

Anexo A: Historia del *Lean Thinking* [11]

Considerando que la agilidad en sí proviene del Lean Thinking, su historia se considera interesante para entender como partió todo.

Piezas Intercambiables

La historia comienza en Francia, antes de la revolución francesa, cuando a Honoré Blanc se le ocurrió que podía construir armas en base a piezas intercambiables. Al mostrar su idea a los diplomáticos y militares, ellos la encontraron genial ya que por primera vez se podrían construir armas de forma masiva, con personal medianamente capacitado usando máquinas sencillas de utilizar con patrones estándar.

Pese a que tomó varios años lograr el objetivo que se buscaba al fabricar las armas con piezas que encajaran fácilmente, se consideró el inicio del llamado "American system of manufacture".

Personas Intercambiables

Con el avance de la historia, en el año 1914, Henry Ford había comenzado con la construcción y ensamblaje del modelo T. Como había avanzado la revolución industrial, Ford ya había cambiado alrededor del 85% del personal por máquinas automatizadas, por lo que consideró sería una buena idea subir el sueldo de los trabajadores que quedaban además de disminuirles las horas de trabajo.

El aumento del sueldo también se podía justificar porque había logrado disminuir los tiempos de ensamblaje de 12 hrs a 90 min, gracias a la mejora en la eficiencia de su línea de producción. Esto lo logró al darse cuenta que un trabajador con un sueldo fijo se esforzaba por trabajar lento ya que no había ningún tipo de incentivo extra para él. Su solución fue dividir la línea de producción en pequeñas etapas y entrenar a los trabajadores para hacer exactamente eso, hacerlo bien y en el tiempo solicitado. Ya que el entrenamiento podía demorar desde 10 min, era fácil tomar a un trabajador e intercambiarlo por otro, en otra parte de la línea de producción.

Pese a que era una buena idea, tenía un problema grave: su sistema sólo servía para un tipo de vehículo, por lo tanto cuando el mercado comenzó a diversificarse, Ford terminó siendo más lento que su competencia, General Motors, quienes había desarrollado un sistema eficiente pensado en varias líneas de producción, cada una para un modelo diferente de automóviles.

Los Toyodas

En el año 1927, Toyoda presentó un telar que funcionaba casi de manera automática, con alta precisión e identificación de errores. El sistema era capaz de repararse sólo y si no podía, entonces se detenía y avisaba al tejedor que estuviera disponible.

Pero llegar a tal logro no fue fácil. El primer telar automático inventado 40 años atrás, era impreciso y tenía un desempeño de bajo nivel en general. Ahí fue donde Sakichi Toyoda contrató al ingeniero americano Charles Francis para instaurar la manufactura americana ("American system of manufacture"). Así es como se construyó un taller especializado para construir telares, con estándares y protocolos, reorganización de la línea de producción, incluyendo piezas intercambiables. Pero la mantención y fabricación de telares necesitaba de personal altamente calificado para lograr la precisión requerida, por lo tanto la política de personas intercambiables, no podía ser aplicada en este caso. Se contrataban sólo profesionales con estudios universitarios y no sólo se dedicaban a fabricar, sino también a investigar, ya que Toyoda tenía un área especialmente dedicada a esto.

Lograron construir el telar que se reparaba sólo, detectaba fallas y se detenía, al igual que podía funcionar toda la noche desatendido. Con las ganancias, el hijo de Sakichi, Kiichiro, decidió iniciarse en la industria de automóviles, de donde nació Toyota (la *d* de Toyoda se cambió por *t*, porque era más fácil de escribir es japonés).

El sistema de producción de Toyota

El sistema que funcionaba en América para el término de la segunda guerra mundial, consistía en ahorrar a través de economías de escala, lo que significaba fabricar cientos de piezas iguales para

construir los vehículos, lo cual no era viable en esa época para una empresa emergente, por los costos implicados, la escasez de material, la demora en las órdenes, etc.

Así fue como Kiichiro pensó en la metodología "Just in Time" ("Justo a tiempo"). Era la única forma de poder ponerse a la par con sus competidores americanos. La metodología consistía en que todas las piezas necesarias para construir tenían que llegar a la línea de producción *Just in time.* Pero esto no era fácil de implementar, ya que en las bodegas las piezas tenían que estar hechas para poder ser enviadas a la línea de ensamblaje cuando fueran necesitadas.

Después de la muerte de Kiichiro, un gerente de taller que había trabajado con los Toyoda desde la época de los telares, Taiichi Ohno, tomó la idea de Kiichiro y consideró ponerla en práctica. Él tenía la experiencia, además de que había estudiado el sistema de Ford y la metodología que utilizaban los supermercados para manejar su inventario.

Pero no fue fácil. Los esfuerzos e ideas de Ohno fueron fuertemente objetados, y no fue sino gracias al apoyo del primo de Kiichiro, Eiji Toyoda, que había estado en cargos gerenciales a lo largo de la compañía, que pudieron salir adelante. Ellos tenían claro que la economía de escala no era la respuesta, ya que pese a que podían tener ahorros de hasta un 25%, los costos aumentaban hasta un 35% cada vez que la variedad se duplicaba. La clave estaba en conquistar la complejidad.

Autonomatización ó Jidoka

Cuando los telares podían funcionar desatendidos, porque cuando detectaban una falla se detenían automáticamente, es representable desde el punto de vista del ser humano. Cuando una persona se quema, automáticamente suelta el objeto caliente, sin que el reflejo pase por el cerebro. Esa era la base de la idea de Ohno, "automatización con un toque humano", cuando en la línea de producción insertó el sistema de detención de la línea, ya que ante cualquier problema la línea debía detenerse y después analizar el problema.

Pilares del sistema de producción de Toyota

Producción sin stock

Una de las claves del Just in Time, era producir sin stock, fabricar sólo lo necesario cuando fuera necesario, para lo cual tenía que contar con máquinas capaces de configurarse para fabricar otra pieza en pocos minutos.

Desde el punto de vista de desarrollo de software, esto tiene que ver con el tiempo de desarrollo de una aplicación. Muchas organizaciones demoran meses en desarrollar una aplicación porque le agregan tantas características y funcionalidades como sea posible en una sola entrega. Esto implica mucho tiempo de entrenamiento, pruebas y trabajo de integración con las demás entregas.

Pero como usuario, por ejemplo, se espera que un antivirus sea actualizado con una entrega bien probada, horas después de que sea lanzada una nueva amenaza. El cambio es tan pequeño que el entrenamiento y la integración no son un problema.

Inspección cero

La idea detrás de la autonomatización es fabricar y diseñar a prueba de errores. No significa un mejor sistema de control de calidad, sino diseñar para que cualquier falla que pudiera suceder, asumirla como que va a ocurrir. El ejemplo es un enchufe para conectar un computador a un monitor. Estos enchufes tienen una posición única, por lo cual es imposible conectarlos al revés o mal, por lo cual no se requiere una inspección para ver si el enchufe está bien conectado.

Just in time

Pese a que el Sistema de Producción de Toyota era conocido en Japón, fue ignorado hasta la crisis del petróleo de 1973, donde Toyota destacó por superar rápidamente las consecuencias de la crisis. En América, el sistema comenzó a ser estudiado a medida que las empresas veían como ingresaban al mercado empresas japonesas con costos muy bajos, lo que requería un mayor estudio de ellas. Así fue como se hizo conocido el sistema Just in Time.

El sistema se puede explicar a través de la Figura A1. El inventario es el nivel del mar. Al disminuir el nivel del mar, van quedando a la vista grandes rocas que representan los problemas que pueden pasar sin ser detectados, como procesos fuera de control, sobre stock, etc. Al quedar a la vista dichas rocas, éstas pueden ser retiradas para evitar que el barco choque con ellas, hasta que el camino quede prácticamente despejado sólo con pequeñas piedrecitas.

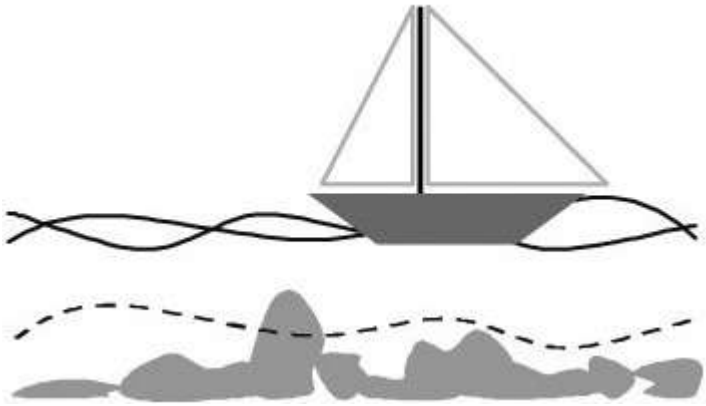

FIGURA A1. Disminución de inventario desde el punto de vista de un barco

La clave es dejar de maximizar la eficiencia local de cada máquina y centrarse en la eficiencia general. La eficiencia local sólo genera tener las máquinas funcionando al 100% produciendo cada vez más inventario hasta no saber qué hacer con él. En las empresas donde se ha aplicado el Just in Time, sorprende como la productividad de la planta aumenta sin tener a las máquinas trabajando al 100%.

Lean o “Sin desperdicios”

En el año 1990, la práctica llamada Just in Time, desarrollada originalmente por Toyota, pasó a llamarse Lean Production (Producción sin desperdicios). Muchas empresas ya reconocían las ventajas del modelo y trataron de incorporarlo a sus propias plantas de producción pero sin los resultados esperados por ellos.

Grandes empresas de muchos años de antigüedad eran reacias a dejar el método tradicional en el que habían invertido tanto dinero y esfuerzo, por lo cual optaban por la opción de aplicar sólo partes del modelo en sus plantas. De esta manera perdían el punto principal del lean thinking:

"*Transferir el máximo número de tareas y responsabilidades a aquellos trabajadores que realmente están agregando valor al carro en la línea y tener un sistema para detectar las fallas, rastrear rápidamente el problema y una vez descubierto, solucionarlo de raíz".*

A pesar de los retos que implica implementar un nuevo paradigma de producción, muchas iniciativas *lean* han sido exitosas y no solamente en el área de la manufactura, sino que se ha expandido a empresas de tipo ventas de retail, procesamiento de órdenes, mantenimiento de aeronaves. Los principios *lean* han sido también incorporados a cadenas de suministros, desarrollo de productos y desarrollo de software.