

KANBAN

Cambio Evolutivo Exitoso Para su Negocio de Tecnología



David J. Anderson

Prólogo de by Donald G. Reinertsen
Traducción de Masa Kevin Maeda

Elogios para *Kanban*

La obra de David con los sistemas de Kanban ha tenido una influencia significativa sobre la manera en que abordó el desarrollo de software y ha cambiado mi forma de pensar acerca de los procesos. En lugar de ver el trabajo con base a historias, puntos y *timeboxes*, ahora veo el Trabajo-en-Progreso, el flujo y la cadencia. Este libro es un éxito para llevar esta perspectiva a un público más amplio y es una lectura obligatoria para quien busque la manera de crear organizaciones de desarrollo exitosas y sustentables.

—**Karl Scotland**
Consultor Senior, EMC Consulting

Es difícil escribir sobre el tema de Kanban debido a que cada implementación es adaptada a su flujo y cuellos de botella específicos; pero David se las arregla para ofrecer un marco teórico sólido y a su vez mantiene una adherencia estricta a la práctica y los resultados del mundo real.

—**Chris Simmons**
Directore de Desarrollo, Sophos

El libro de Kanban de David Anderson va más allá del nivel introductorio sobre como Kanban dirige el cambio y proporciona una explicación clara de las tuercas y tornillos, dando ejemplos ricos y consejos prácticos. Kanban para el trabajo del conocimiento le da un gran apoyo a la tendencia emergente de autonomía en el lugar de trabajo. Uno de los desarrollos más excitantes de gestión de nuestro tiempo.

—**Christina Skaskiw**
Couch Ágil

La mejor metodología nueva de cambio que he visto para software en los últimos diez años.

—**David A. Bulkin**
Vicepresidente, Lithespeed, LLC

Kanban

*Cambio Evolutivo Exitoso Para su
Negocio de Tecnología*

David J. Anderson
Traducción al español por Masa K Maeda



Sequim, Washington



Blue Hole Press
72 Buckhorn Road
Sequim, WA 98382
www.blueholepress.com

Derechos Reservados © 2010 David J. Anderson
dja@agilemanagement.net

Traducción al español por Masa K Maeda
masa@shojiki-solutions.com

Todos los derechos reservados. Ninguna parte de ésta publicación puede ser reproducida o transmitida en forma alguna o por medio alguno, electrónico o mecánico, incluyendo fotocopia, grabado, o cualquier sistema de almacenamiento o recuperación sin permiso escrito del publicista.

Impreso en los Estados Unidos de America.

Datos de aplicaciónn del *Library of Congress Catalog-in-Publication*
disponibles en www.loc.gov

ISBN: 978-0-9845214-3-2 (español)

Derechos del arte de la portada © 2010 por Pujan Roka
Derechos de la foto de la portada © 2010 por Laurence Cohen
Diseño de la portada y del interior por Vicki L. Rowland
Fotos en la introducción del Capítulo 2 utilizadas con permiso, Thomas Blomseth

10 9 8 7 6 5 4 3

❖ TABLA DE CONTENIDO ❖

Prólogo de la versión en español xvii

❖ PARTE UNO ❖

Introducción 1

CAPÍTULO 1

Resolviendo el dilema de un director ágil 3

 Mi búsqueda por el ritmo sustentable 4

 Mi búsqueda por la gestión de cambio exitosa 5

 De Drum-Buffer-Rope a kanban 8

 Surgimiento del método kanban 9

 Adopción de Kanban por la comunidad 9

 El valor de kanban es contra-intuitivo 10

CAPÍTULO 2

¿Qué es el método Kanban? 13

 ¿Qué es un sistema kanban? 15

 Kanban aplicado en el desarrollo de software 15

 ¿Por qué usar un sistema kanban? 17

 Kanban, un Sistema Adaptivo Complejo para Lean 17

 Comportamiento surgidos con Kanban 18

 Kanban como un otorgador de permisos 19

❖ PARTE DOS ❖

Beneficios de Kanban 23

CAPÍTULO 3

Una receta para el éxito 25

 Implementando la receta 26

Enfoque en calidad _____	27
Reduzca el Trabajo-en-Progreso y entregue seguido _____	29
Equilibrio de la demanda contra el throughput _____	35
Priorizar _____	36
Ataque las fuentes de variabilidad y mejore el proceso _____	37
Receta para el éxito y Kanban _____	38
 CAPÍTULO 4	
Del peor al mejor en cinco cuartos_____	41
El Problema _____	42
Visualice el flujo de trabajo _____	43
Factores que afectan el rendimiento _____	44
Hacer las políticas de proceso explícitas _____	45
La estimación era un desperdicio _____	46
Limitar el trabajo-en-progreso _____	46
Estableciendo una cadencia de entrada _____	46
Obteniendo una nueva ganga _____	48
Implementando los cambios _____	49
Ajustando políticas _____	49
Buscando por más mejoras _____	50
Resultados _____	52
 CAPÍTULO 5	
Una cultura	
de mejora continua_____	55
La Cultura Kaizen _____	56
Kanban acelera la capacidad y la madurez organizacional _____	56
Cambio Sociológico _____	61
El cambio cultural es quizás el mayor beneficio de Kanban _____	65
 ❖ PARTE TRES ❖	
Implementando Kanban _____	67
 CAPÍTULO 6	
Mapeo de la cadena de valor _____	69
Definiendo un punto de inicio y de fin para el control _____	70
Tipos de ítems de trabajo _____	70

Trazando una pared de tarjetas	71
Análisis de demanda	75
Asignando capacidad de acuerdo a la demanda	76
Anatomía de una tarjeta de ítem de trabajo	77
Seguimiento electrónico	78
Estableciendo los límites de entrada y salida	79
Haciendo frente a la concurrencia	80
Haciendo frente a actividades desordenadas	82
CAPÍTULO 7	
Coordinación con	
sistemas Kanban	85
Control visual y arrastre	85
Seguimiento electrónico	87
Reuniones de pie diarias	88
La pos-reunión	89
Reuniones de Reposición de Cola	90
Reuniones de planificación de entrega	91
Triaje	92
Revisión y escalamiento del registro de asuntos	93
Sticky friend	94
Sincronización a través de localidades geográficas	95
CAPÍTULO 8	
Estableciendo una cadencia de entrega	97
Costos de coordinación de entrega	99
Costos de transacción de entrega	100
La eficiencia de la entrega	102
Acordando con una cadencia de entrega	103
Mejore la eficiencia para incrementar la cadencia de entrega	103
Haciendo entregas bajo demanda o ad hoc	104
CAPÍTULO 9	
Estableciendo una cadencia de entrada	109
Costos de coordinación de priorización	109
Acordando en una cadencia de priorización	111
Eficiencia de la priorización	112

Costos de transacción de la priorización _____	113
Mejore la eficiencia para incrementar la cadencia de la priorización _____	113
Efectuando priorización bajo demanda o ad hoc_____	114

CAPÍTULO 10

Estableciendo los límites del Trabajo-en-Progreso _____	119
Límites para las tareas de trabajo _____	119
Límites para las Colas _____	121
Buffers de cuellos de botella_____	122
Tamaño de la cola de entrada_____	122
Secciones ilimitadas de flujo de trabajo _____	124
No estrese a su organización _____	125
No establecer un límite de WIP es un error _____	125
Asignación de la capacidad _____	126

CAPÍTULO 11

Estableciendo los límites del Trabajo-en-Progreso _____	129
Definiciones de las clases de servicio habituales _____	130
Expreso _____	131
Fecha de entrega fija _____	132
Clase estándar _____	133
Clase intangible _____	134
Políticas para las clases de servicio _____	135
Políticas expreso _____	135
Políticas de fecha de entrega fija _____	136
Políticas de clase estándar _____	136
Clase intangible _____	137
Determinando un objetivo para la entrega de servicio _____	138
Asignando una clase de servicio _____	139
Poniendo en uso las clases de servicio _____	140
Asigne capacidad a las clases de servicio _____	140

CAPÍTULO 12

Métricas y reportes de gestión _____	145
Rastreando el WIP _____	146
Tiempo de entrega _____	146

Rendimiento de la fecha límite	148
Throughput	148
Asuntos e ítems de trabajo bloqueados	149
Eficiencia de flujo	150
Calidad inicial	151
Carga de falla	152
 CAPÍTULO 13	
Escalando Kanban con sistemas de dos niveles	155
Requerimientos jerárquicos	156
Desacople la entrega de valor de la variabilidad del ítem de trabajo	157
Paredes de tarjetas de dos niveles	159
Introduciendo carriles de nado	160
Enfoque alternativo al tamaño de la variabilidad	161
Incorporando clases de servicio	161
Integración de sistemas	162
Gestión de recursos compartidos	162
 CAPÍTULO 14	
Revisión de operaciones	165
La reunión previa	165
Establezca un tono de negocio desde el principio	166
Tener invitados amplía la audiencia y agrega valor	166
Agenda principal	167
La clave de la transición lean	168
La cadencia apropiada	168
Demostrando el valor de los directores	169
El enfoque organizacional fomenta kaizen	170
Un ejemplo anterior	170
 CAPÍTULO 15	
Comenzando una iniciativa Kanban de cambio	173
Kanban acelera la madurez y capacidad organizacional	173
La meta principal de nuestro sistema kanban	175
Las metas secundarias de nuestro sistema kanban	175
Conozca las metas y articule los beneficios	181
Pasos para comenzar	181

Kanban impacta con un tipo distinto de ganga _____	183
Logrando la ganga de Kanban _____	185
Límites de WIP _____	185
Priorización_____	187
Entrega/entrega _____	187
Tiempos de entrega y clases de servicio _____	188

❖ PARTE CUATRO ❖

Haciendo Mejoras _____	193
-------------------------------	------------

CAPÍTULO 16

Tres tipos de oportunidad de mejora _____	195
Cuellos de botella, eliminación de desperdicio y reducción de variabilidad	196
Teoría de Restricciones _____	196
Lean, TPS y reducción de desperdicio _____	198
Deming y seis sigma _____	199
Ajustando Kanban a la cultura de su empresa _____	201

CAPÍTULO 17

Cuellos de botella y disponibilidad instantánea _____	203
Recursos de capacidad restringida _____	205
Acciones de elevación _____	205
Acciones de explotación/protección _____	206
Acciones de subordinación _____	209
Recursos de disponibilidad no-instantánea _____	210
Acciones de explotación/protección _____	212
Acciones de subordinación _____	213
Acciones de elevación _____	214

CAPÍTULO 18**Un modelo**

económico para lean _____	217
Redefiniendo “desperdicio” _____	217
Costos de transacción _____	218
Costos de coordinación _____	220
¿Cómo sabe si una actividad es un costo? _____	222

Carga de falla _____	222
CAPÍTULO 19	
Fuentes de variabilidad _____	225
Fuentes internas de variabilidad _____	227
Tamaño del ítem de trabajo _____	227
Mezcla de tipos de ítem de trabajo _____	228
Mezcla de clases de servicio _____	229
Flujo irregular _____	230
Re-trabajo _____	231
Fuentes Externas de Variabilidad _____	231
Ambigüedad de Requerimientos _____	231
Solicitudes Expreso _____	233
Flujo Irregular _____	234
Disponibilidad del Entorno _____	235
Otros Factores de Mercado _____	236
Dificultad Programando Actividades de Coordinación _____	237
CAPÍTULO 20	
Políticas de gestión de asuntos y escalamiento _____	241
Gestionando Asuntos _____	242
Escalando Asuntos _____	243
Rastreando y Reportando Asuntos _____	244
CAPÍTULO 21	
Kanban en Latinoamérica	
por Masa K Maeda _____	247
Kanban en México _____	247
Un Caso a Modo de Ejemplo _____	249
Kanban en el resto de Sudamérica _____	250
Notas _____	253
Reconocimientos _____	255
Reconocimientos de la versión en español _____	258
Acerca del autor _____	259
Recursos adicionales sobre Kanban _____	260
Índex _____	261

❖ *Dedicado a Nicola and Natalie* ❖

❖ Prólogo ❖

Siempre le presto atención a la obra de David Anderson. Mi primer contacto con él fue en octubre de 2003, cuando me envió una copia de su libro, *Agile Management for Software Engineering: Applying Theory of Constraints for Business Results*. Como su título lo indica, este libro fue altamente influenciado por la Teoría de Restricciones (TDR; en inglés *Theory-of-Constraints—TOC*) de Eli Goldratt. Más adelante, en marzo de 2005, lo visité en Microsoft cuando él estaba ya haciendo un trabajo impresionante con los diagramas de flujo acumulativo. Algún tiempo después, en abril de 2007, tuve la oportunidad de observar su implementación de frontera del sistema kanban en Corbis.

Trazo esta cronología para darle a usted un sentido del ritmo implacable al cual el pensamiento de gestión de David ha avanzado. Él no se queda bloqueado en una sola idea ni trata de forzar al mundo a que se adapte a ella. En cambio, le presta especial atención al problema global que está tratando de resolver, se mantiene abierto a las distintas soluciones posibles, las prueba en la acción y reflexiona sobre por qué funcionan. Usted verá los resultados de ese enfoque en este nuevo libro.

Por supuesto, la velocidad es más útil si está en la dirección correcta y estoy seguro de que David va en tal dirección. Estoy particularmente contento por este último trabajo con los sistemas kanban. Siempre he encontrado las ideas de la manufactura Lean más directamente útiles en el desarrollo de productos que los de la TDR. De hecho, en octubre de 2003 le escribí a David diciéndole: “Una de las grandes debilidades de la TDR es su falta de énfasis en la importancia del tamaño del lote

(*batch-size*). Si la primera prioridad es encontrar y reducir la restricción entonces a menudo se estará resolviendo el problema equivocado". Todavía creo que esto es cierto.

Durante nuestra reunión en 2005, alenté a David nuevamente a observar más allá del enfoque de cuello de botella de la TDR. Le expliqué que el éxito espectacular del Sistema de Producción de Toyota (TPS) no tenía nada que ver con encontrar y eliminar los cuellos de botella. Las mejoras de rendimiento de Toyota se debieron a la reducción del tamaño del lote y a la reducción de la variabilidad para reducir el inventario de trabajo-en-progreso (en inglés *work-in-progress – WIP*). Los beneficios económicos surgieron a partir de la reducción de inventario; y fueron los sistemas para limitar WIP, tales como kanban, los que hicieron esto posible.

Cuando visité Corbis en 2007 observé una implementación impresionante de un sistema kanban. Le indiqué a David que él logró superar el enfoque de kanban mucho más allá de como es utilizado por Toyota. ¿Por qué le indiqué esto? El Sistema de Producción Toyota está elegantemente optimizado para lidar con tareas repetitivas y predecibles: tareas con duración homogénea y costos de demora homogéneas. Bajo tales condiciones, es correcto utilizar enfoques tales como priorización tipo primero-en-entrar-primer-en-salir (FIFO). Es también correcto bloquear la entrada de trabajo cuando el límite del WIP es alcanzado. Sin embargo, estos enfoques no son óptimos cuando tenemos que lidar con trabajos no repetitivos e impredecibles y con diferente costo de retraso—exactamente con lo que tenemos que lidar dentro del desarrollo de productos. Necesitamos sistemas más avanzados y este libro es el primero que describe estos sistemas a un nivel de detalle práctico.

Me gustaría ofrecer algunas advertencias breves a los lectores. En primer lugar, si usted cree que ya entiende como funcionan los sistemas kanban, probablemente está pensando en los sistemas kanban utilizados en la manufactura Lean. Las ideas en este libro van mucho más allá de tales sistemas simples que utilizan límites estáticos del WIP, itinerarios de FIFO y una sola clase de servicio. Preste mucha atención a estas diferencias.

En segundo lugar, no piense sobre este enfoque solamente como un sistema de control visual. La manera en que los pizarrones de kanban hacen el WIP visible es sorprendente, pero eso es sólo un pequeño aspecto de este enfoque. Si usted lee este libro cuidadosamente encontrará que suceden muchas más cosas. La perspicacia real está en aspectos tales como el diseño de los procesos de llegada y salida, la gestión de los recursos no fungibles y el uso de las clases de servicio. No se distraiga por la parte visual y permita que se pierdan las sutilezas.

En tercer lugar, no descarte estos métodos porque parezcan fáciles de usar. Esta facilidad de uso es un resultado directo de la visión de David en lo que produce el máximo beneficio con el mínimo esfuerzo. Él está muy consciente de las necesidades de los practicantes y ha prestado mucha atención en lo que realmente funciona. Métodos sencillos crean el menor trastorno posible y casi siempre producen los mayores beneficios sustentables.

Este es un libro interesante e importante que merece ser leído cuidadosamente. Lo que obtendrá de él dependerá de la seriedad con que lo lea. Ningún otro libro lo expondrá más a estas ideas avanzadas. Espero que lo disfruten tanto como yo.

Don Reinertsen,
Febrero 7, 2010
Redondo Beach, California
Autor de *The Principles of Product Development Flow*

Prólogo de la versión en español

Mi primer contacto con Lean sucedió durante mis años de vida en Japón como estudiante de posgrado cuando tuve la oportunidad de visitar una planta de automóviles de Toyota y también trabajé para Justsystems Corporation, la empresa de software número uno en Japón en aquel entonces. Aprendí los beneficios del énfasis en la alta calidad y en el trabajo colaborativo. Una vez establecido en los Estados Unidos, específicamente en Cupertino, California, tuve la oportunidad de conocer y trabajar con Brian Marick con quién llevé algunas conversaciones sobre lo problemático de las metodologías de desarrollo de software. Brian incrementó mi pasión por la calidad y por la resistencia a todo status-quo ineficiente (gracias Brian). Cuando el Agile Manifiesto se hizo público en 2001 me llevé la agradable sorpresa de ver a Brian como uno de sus formadores. Profesionalmente confronté retos para inyectar *lean* y *agile* debido a la resistencia hacia ellas de las empresas en esos días. Curiosamente mis éxitos se lograron casi siempre sin usar las palabras consideradas tabú en algunas empresas: *Lean* o *Agile*.

Estando enteramente convencido del tremendo impacto positivo que *lean* y *agile* pueden tener decidí en Septiembre de 2008 crear mi propia firma dedicada plena y explícitamente a inyectar *lean-agile* en las empresas. Esa jornada y mi alto interés en *lean* me llevó a compenetrarme en el mundo de Kanban.

Haber conocido a David J. Anderson, aprender Kanban directamente de él, y formar parte de la comunidad de Kanban ha sido una de las mejores experiencias de mi vida profesional. Cuando David ofreció acceso a su manuscrito preliminar para revisarlo previo a su publicación brinqué a la oportunidad a modo de reflejo; y cuando el libro estaba ya por publicarse intercambié correos electrónicos ofreciéndole traducirlo al español sin tener una medida realista de la magnitud del reto. David aceptó de inmediato. Durante el tiempo que he conocido a David e interactuado con él he aprendido muchísimo mas sobre Kanban y *lean* de

lo que hubiera imaginado. David es un excelente profesionista con una variedad y profundidad de conocimientos admirable, los cuales siempre está dispuesto a compartir con una actitud positiva y de manera amena. Igualmente importante es su interés genuino en el factor humano tanto como en la mejora de las organizaciones. Ese modo de ser se refleja directamente en la manera en que Kanban ha evolucionado como base cultural empresarial y como metodología. No es sorprendente ver al influencia que Kanban está teniendo ya fuera de las áreas de desarrollo de software, tales como otras áreas de las empresas, en educación y en otras disciplinas.

La traducción a sido una experiencia única. Esta es la primera vez que efectúo tal labor y como resultado mi apreciación y respeto por toda persona que ha traducido libros ha crecido significativamente. Confronté varios retos: Asegurarme que mi entendimiento de los conceptos y conocimiento no tan solo es correcto sino también transmitido acertadamente; respetar el estilo de David para asegurarme que el libro sigue siendo suyo y no una especie de “versión Masa” del libro; evitar que el tipo de español utilizado no sea localizado sino lo mas genérico posible. A pesar de mis esfuerzos asumo la responsabilidad por los errores que puedan existir y me disculpo por ello. Al mismo tiempo, correcciones son factibles y sus comentarios para la mejora de éste manuscrito serán altamente apreciados.

Espero que encuentre este libro sobre Kanban altamente disfrutable, fascinante, y valioso. Así mismo, deseo que le sirva como una base de transformación significativa e importante para su organización. Le invito también a que participe activamente en la comunidad de Kanban a nivel mundial.

Masa Kevin Maeda
Agosto 20, 2010
Campbell, California
CEO y fundador, Shojiki Solutions

❖ PARTE UNO ❖

INTRODUCCIÓN

❖ CAPÍTULO 1 ❖

Resolviendo el dilema de un director ágil

En 2002 yo era un director de desarrollo asediado en un lugar remoto de la división de PCS (teléfonos móviles) de Motorola con sede en Seattle, Washington. Mi departamento era parte de una nueva empresa que Motorola había adquirido un año antes. Desarrollábamos software para servidores de red para servicios de datos inalámbricos tales como descarga por aire y administración de dispositivos por aire. Estas aplicaciones de servidor eran parte de sistemas integrados que trabajaban mano a mano con el código de cliente en los teléfonos móviles, así como con otros elementos dentro de las redes de las compañías de telecomunicaciones y la infraestructura de back-office, tales como la facturación. Nuestras fechas de finalización eran determinadas por los directores sin considerar la complejidad de ingeniería, el riesgo o el tamaño del proyecto. Nuestro código base había evolucionado desde la creación de la compañía, donde se habían tomado muchos atajos. Un desarrollador senior insistió que nos refiriéramos a nuestro producto como “un prototipo”. Teníamos una necesidad desesperada de incrementar la productividad y mejorar la calidad a fin de satisfacer las demandas empresariales.

Durante mi trabajo diario, en 2002 y a través de mis esfuerzos de autor, en mi libro anterior¹ mantenía en mente dos desafíos principales. En primer lugar, ¿cómo podría yo proteger a mi equipo de las incesantes demandas del negocio y lograr lo que la comunidad ágil ahora llama “un ritmo sustentable”? Y en segundo lugar, ¿cómo podría yo escalar

exitosamente la adopción de un enfoque ágil en una empresa y superar la inevitable resistencia al cambio?

Mi búsqueda por el ritmo sustentable

En 2002, la comunidad ágil hacía referencia a “la semana de 40 horas”² como la noción de ritmo sustentable. Los principios detrás del Manifiesto Ágil³ nos dicen que “Los procesos ágiles promueven el desarrollo sustentable. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante indefinidamente”. Dos años antes, mi equipo en Sprint PCS se había acostumbrado a que yo les dijera que “el desarrollo de software a gran escala es una maratón, no un sprint”. Si los miembros del equipo iban a mantener el ritmo a largo plazo en un proyecto de 18 meses, entonces no podíamos permitirnos el lujo de desgastarlos al cabo de uno o dos meses. El proyecto tenía que ser planificado, presupuestado, programado y estimado, para que los miembros del equipo pudieran trabajar un número de horas razonables diariamente y evitaran cansarse. El reto para mí, como director, era alcanzar este objetivo y acomodar todas las demandas del negocio.

En mi primer trabajo como gestor en 1991, en una nueva empresa de 5 años de antigüedad que hacía tabletas para captura de vídeo para PCs y otras computadoras pequeñas, la retroalimentación del Director General era que la dirección me veía como “muy negativo”. Siempre decía que “no” cuando me pedían aún más productos o funcionalidades y nuestra capacidad de desarrollo ya se había estrechado al máximo. Para 2002 ya había un patrón claro: me había pasado más de 10 años diciendo “No”, empujando en contra de las constantes y cambiantes demandas de los propietarios de negocios.

En general los equipos de ingeniería de software y los departamentos de TI parecían estar a la merced de otros grupos que negociaban, persuadían, intimidaban y engatusaban incluso los planes más defendibles y derivados objetivamente. Aún los planes basados en un análisis minucioso y respaldados por años de datos históricos eran vulnerables. La mayoría de los equipos, los cuales no tenían un solo método de análisis completo o cualquier otro dato histórico, estaban indefensos en manos de quienes los llevaban a comprometerse con entregables desconocidos (y a menudo completamente irrazonables).

Mientras tanto la fuerza de trabajo había llegado a aceptar como norma los horarios locos y los compromisos de trabajo ridículos. Aparentemente se supone que los ingenieros de software no tienen una vida social o familiar. ¡Si eso se siente como un abuso, es porque de hecho lo es! Conozco a demasiadas personas cuyo compromiso con el trabajo ha dañado la relación con sus hijos y con otros miembros de su familia de manera irreparable. Es difícil tener simpatía con el habitual *geek* de desarrollo de software. En el estado de Washington, en los Estados Unidos, donde radico, los ingenieros de software están en segundo lugar (después de los dentistas) en cuanto a nivel de remuneración anual. Como trabajadores en la línea de ensamblaje de Ford durante la segunda década del siglo 20—ganando

cinco veces el salario medio en los Estados Unidos—nadie se preocupó por la monotonía del trabajo, o por el bienestar de los trabajadores porque estaban muy bien remunerados. Es difícil imaginar el surgimiento de la labor organizada en las áreas de trabajo del conocimiento tales como el desarrollo de software; principalmente porque es difícil imaginar a alguien que haga frente a las causas raíz de los problemas físicos y psicológicos que los desarrolladores sufren cotidianamente. Los empresarios más ricos han sido aptos en agregar beneficios adicionales para la atención de la salud tales como: masajes, psicoterapia y ocasionalmente días de descanso por “salud mental” en lugar de perseguir las causas raíz del problema. Un escritor técnico en una empresa prestigiosa de software me comentó en una ocasión que “¡no existe estigma por el uso de antidepresivos, todos los están usando!” Como respuesta a este abuso, los ingenieros de software tienden a aceptar las demandas, cobrando sus sueldos de lujo y sufriendo las consecuencias.

Yo quería romper ese molde. Yo quería encontrar un enfoque en el que todos ganasen; que me permitiera decir “Sí”; que me permitiera proteger al equipo y hacer posible un ritmo sustentable. Quería corresponder a mi equipo—devolverles su vida social y familiar—y mejorar las condiciones que causaban problemas de salud relacionadas con el estrés en los desarrolladores jóvenes como de veintitantos. Por lo tanto decidí tomar las riendas e intentar hacer algo acerca de estos problemas.

Mi búsqueda por la gestión de cambio exitosa

La segunda cosa en mi mente era el reto de ser líder del cambio en grandes organizaciones. Fui director de desarrollo para Sprint PCS y después para Motorola. En ambas empresas había una necesidad real a nivel de negocio de desarrollar una manera más ágil de trabajar. Pero en ambos casos luché para llegar a escalar los métodos ágiles a más de uno o dos equipos.

No tuve suficiente poder jerárquico en ambos casos como para imponer un cambio en un gran número de equipos. Estaba intentando influir en el cambio bajo la petición de la alta dirección, pero sin estar en ninguna posición de poder. Me habían pedido influir en mis compañeros para que hicieran cambios similares en sus equipos a los que yo había puesto en práctica en el mío. Los otros equipos se resistieron a adoptar las técnicas que claramente estaban dando mejores resultados en mi equipo. Probablemente hubo muchas facetas de esta resistencia, pero el tema más común fue que la situación de cada equipo era diferente; las técnicas de mi equipo tendrían que ser modificadas y adaptadas a las necesidades específicas de los otros equipos. A mediados de 2002 llegué a la conclusión de que intentar hacer cumplir un proceso prescriptivo de desarrollo de software en un equipo no funciona.

Se necesitaba adaptar un proceso para cada situación específica. Pero para poder lograrlo se necesitaba un liderazgo activo en cada equipo, lo cual faltaba a menudo. Aún con

el liderazgo correcto dudé que un cambio significativo pudiera ocurrir sin un marco establecido y sin una guía sobre cómo adaptar el proceso para ajustarse a diferentes situaciones. Sin esta guía para el líder, el *coach*, o el ingeniero de proceso, era probable que cualquier adaptación fuera impuesta de manera subjetiva, basada en creencias supersticiosas. Esto tenía tanta probabilidad de generar tajadas y objeciones como lo era la imposición de la plantilla de un proceso inadecuado.

Me propuse parcialmente a tratar esta cuestión en el libro que estaba escribiendo en ese entonces: *Agile Management for Software Engineering*. Yo cuestionaba: “Por qué el desarrollo ágil produce mejores resultados económicos que los enfoques tradicionales?”

Busqué utilizar el marco de la Teoría de Restricciones⁴ para construir el caso.

Mientras investigaba y escribía ese libro me di cuenta de que de alguna manera, cada situación es única. ¿Por qué tenía que estar el factor limitante o cuello de botella en el mismo lugar para todo equipo y todo proyecto en toda ocasión? Cada equipo es diferente: diferente conjunto de habilidades, capacidades y experiencia. Cada proyecto es diferente: diferente presupuesto, calendario, alcance y perfil de riesgo. Y cada organización es diferente: una cadena de valores distinta operando en un mercado diferente, como se muestra en la Figura 1.1. Se me ocurrió que esto podría darme una pista sobre la resistencia al cambio. Si las propuestas para cambios de prácticas laborales y de conductas no tienen un beneficio percibido, la gente se va a resistir. Si esos cambios no afectan lo que los miembros del equipo perciben como su restricción o factor limitante, entonces se resistirán. En pocas palabras, los cambios sugeridos fuera de contexto serán rechazados por los trabajadores que viven y entienden el contexto del proyecto.



Figura 1.1 Por qué las metodologías de desarrollo de “una misma medida para todos” no funcionan

Parecía mejor dejar que un nuevo proceso evolucionase mediante la eliminación de un cuello de botella tras otro. Esta es la tesis central de la *Teoría de Restricciones* de Goldratt. Sin dejar de reconocer que yo tenía mucho que aprender, sentía que había valor en el material y continué adelante con el manuscrito previsto originalmente. Sabía muy bien que mi libro no daba consejos sobre cómo implementar las ideas a gran escala debido a que ofrecía poco o ningún consejo sobre la gestión de cambio.

El enfoque de Goldratt, explicado en el capítulo 16, trata de identificar un cuello de botella y después encontrar las maneras de aliviarlo hasta que no límite el rendimiento. Una vez que esto sucede, un cuello de botella nuevo emerge y se repite el ciclo. Es un enfoque iterativo para mejorar el rendimiento de forma sistemática, identificando y eliminando cuellos de botella.

Me di cuenta de que podía sintetizar esta técnica con algunas ideas de Lean. Modelando el flujo de trabajo de un ciclo de vida de desarrollo de software como una cadena de valor y después creando un sistema de seguimiento y visualización para rastrear los cambios de estado de trabajo emergente conforme “fluyera” a través del sistema, pude ver los cuellos de botella. La capacidad de identificar cuellos de botella es el primer paso en el modelo subyacente de la Teoría de Restricciones. Goldratt ya había desarrollado una aplicación de la teoría de problemas de flujo, extrañamente llamada *Drum-Buffer-Rope*. Independientemente de lo extraño de su nombre, me di cuenta de que una solución simplificada del *Drum-Buffer-Rope* podría utilizarse para el desarrollo de software.

Genéricamente, el *Drum-Buffer-Rope* es un ejemplo de una clase de soluciones conocidas como sistemas de arrastre (en inglés: *pull system*). Como veremos en el capítulo 2, un sistema kanban es otro ejemplo de sistema de arrastre. Un efecto secundario interesante de los sistemas de arrastre es que limitan el trabajo-en-progreso (WIP; en inglés *WIP*) a una cantidad acordada, con lo que impiden que los trabajadores se saturén. Además, sólo los trabajadores en la estación donde se tiene un cuello de botella permanecen sobrecargados, todos los demás tienen tiempo de holgura. Me di cuenta de que los sistemas de arrastre podrían resolver mis dos desafíos. Un sistema de arrastre me permitiría: implementar un proceso de cambio de manera incremental, reducir (esperanzadoramente) la resistencia al cambio de manera significativa y facilitar un ritmo sustentable. Determiné establecer un sistema de arrastre de *Drum-Buffer-Rope* a la primera oportunidad. Quería experimentar con la evolución del proceso incremental y ver si con ello creaba un ritmo sustentable y la disminuía la resistencia al cambio.

La oportunidad llegó en el otoño de 2004 en Microsoft y está documentada en el caso de estudio en el capítulo 4.

De Drum-Buffer-Rope a kanban

La implementación de una solución de *Drum-Buffer-Rope* en Microsoft funcionó bien. Con muy poca resistencia, la productividad incrementó más del triple y los plazos de entrega se redujeron 90%, mientras que el riesgo se redujo en gran medida mediante la mejora de predicciones. En el otoño de 2005 reporté los resultados en una conferencia en Barcelona y nuevamente en el invierno de 2006. Mi trabajo llamó la atención de Donald Reinertsen, quien hizo un viaje especialmente para visitarme en mi oficina en Redmond, Washington. Él quería convencerme de que yo tenía todas las piezas en su lugar para implementar un sistema kanban completo.

“Kan-ban” es una palabra japonesa que literalmente significa “letrero” o “tarjeta” en español. En un ambiente de manufactura, esta tarjeta se utiliza para indicar a una fase anterior del proceso que produzca más. A los trabajadores en cada paso del proceso no se les permite trabajar a menos que se les indique lo contrario con un kanban proveniente de una fase posterior. Mientras que yo estaba consciente de este mecanismo, no estaba convencido de que fuera una técnica útil o una técnica viable para su aplicación en el trabajo del conocimiento, particularmente en la ingeniería de software. Entendía que los sistemas kanban permitían el ritmo sustentable. Sin embargo, yo no tenía conciencia de su reputación como un método para guiar la mejora de procesos incrementales. Yo no sabía que Taiichi Ohno, uno de los creadores del Sistema de Producción Toyota, había dicho: “Los dos pilares del sistema de producción Toyota son: just-in-time y la automatización con un toque humano, o Autonomatización (en inglés: *Autonomation*). La herramienta utilizada para operar el sistema es kanban”. En otras palabras, kanban es fundamental para el kaizen (o “mejora continua”) utilizado en Toyota. Es el mecanismo que lo hace funcionar. He llegado a reconocer esto como una verdad completa a través de mis experiencias a lo largo de cinco años desde ese entonces.

Afortunadamente, Don me presentó un argumento convincente de que debería cambiar de una implementación de *Drum-Buffer-Rope* a un sistema kanban por la razón esotérica de que un sistema kanban se recupera con más tacto de una falla en la estación donde se encuentra el cuello de botella comparado con *Drum-Buffer-Rope*. La comprensión de esta idiosincrasia no es importante para nuestra comprensión y lectura de este libro.

Revisando la solución final implementada en Microsoft, me di cuenta de que, de haberla concebido como un sistema kanban desde el principio, el resultado habría sido idéntico. Fue interesante para mí que dos enfoques diferentes obtuvieran el mismo resultado. Si el proceso resultante es el mismo, no me siento obligado a pensar sobre él específicamente como una implementación tipo *Drum-Buffer-Rope*.

He desarrollado mayor preferencia por el término “kanban” que por el de *Drum-Buffer-Rope*. Kanban se utiliza en la manufactura lean (o el Sistema de Producción Toyota). Este cuerpo de conocimiento tiene mucho más amplia adopción y aceptación que la Teoría de

Restricciones. El término “kanban”, aunque es japonés, es menos metafórico que Drum-Buffer-Rope. Kanban es más fácil de decir, más fácil de explicar y resulta ser más fácil de enseñar e implementar, por lo que es el término que se ha quedado.

Surgimiento del método kanban

En septiembre de 2006, me fui de Microsoft para hacerme cargo del departamento de ingeniería de software en Corbis, un negocio privado para fotografía y propiedad intelectual con base en el centro de Seattle. Alentado por los resultados en Microsoft, decidí implementar un sistema de arrastre kanban en Corbis. Una vez más los resultados fueron alentadores y llevaron al desarrollo de la mayoría de las ideas presentadas en este libro. Este conjunto ampliado de ideas incluye la visualización de flujo de trabajo, tipos de ítems de trabajo, la cadencia, las clases de servicio, reportes de gestión específicos y revisiones de operaciones que definen el proceso de kanban explicado en estas páginas.

En el resto de este libro Kanban (K mayúscula) se refiere al método de cambio evolutivo que utiliza un sistema de arrastre kanban (k minúscula), visualización y otras herramientas descritas en este texto, para catalizar la introducción de ideas lean dentro del desarrollo de tecnología y operaciones de TI. El proceso es evolutivo e incremental. Kanban le permite lograr una optimización de procesos de contexto específico con mínima resistencia al cambio, manteniendo un ritmo sustentable para los trabajadores involucrados.

Adopción de Kanban por la comunidad

En mayo de 2007, Rick Garber y yo presentamos los primeros resultados de Corbis en la conferencia *Lean New Product Development* en Chicago, con una audiencia de alrededor de 55 personas. Ese mismo verano, en la conferencia *Agile 2007* en Washington D.C., llevé a cabo una sesión de open-space para discutir los sistemas kanban. Cerca de 25 personas asistieron. Dos días después, uno de los asistentes, Arlo Belshee, dio una conferencia deslumbrante en la que compartió su técnica de *Planificación Desnuda*⁶. Al parecer, otras personas también habían estado implementando sistemas de arrastre. Un grupo de discusión Yahoo! se formó y creció rápidamente a 100 miembros. En el momento de escribir este libro, el grupo tiene más de 1000 miembros. Varios de los asistentes a la sesión de open-space se comprometieron a probar Kanban en su lugar de trabajo, a menudo con los equipos que han tenido dificultades con Scrum. Los más notables de los adoptantes iniciales fueron: Karl Escocia, Aaron Sanders y Arnold Joe, todos de Yahoo! quienes rápidamente llevaron Kanban a más de 10 equipos en 3 continentes. Otro asistente notable en el open-space fue Kenji Hiranabe quien había estado desarrollando soluciones Kanban en Japón. Poco después escribió dos artículos sobre el tema para InfoQ, atrayendo mucho interés

y atención. En el otoño de 2007 Sanjiv Augustine, autor de *Managing Agile Projects* y uno de los fundadores del *Agile Project Leadership Network* (APLN), visitó Corbis en Seattle y describió nuestro sistema kanban como “el primer método ágil nuevo que he visto en cinco años”.

Al año siguiente, en *Agile 2008*, en Toronto, hubo 6 presentaciones sobre el uso de soluciones Kanban en diferentes escenarios, incluyendo uno de Joshua Kerievsky, de Industrial Logic, una empresa de consultoría y entrenamiento sobre Programación Extrema, mostró como él había desarrollado similares ideas para adaptar y mejorar la Programación Extrema a su contexto de negocios. Ese año, el Agile Alliance galardonó con el Premio Gordon Pask a Arlo Belshee y Kenji Hiranabe por sus contribuciones a la comunidad Agile. Ambos hicieron una contribución notoria sobre el surgimiento de Kanban. Los dos habían producido y comunicado ideas muy similares.

El valor de kanban es contra-intuitivo

En muchos sentidos, el trabajo del conocimiento es la antítesis de una actividad de producción repetitiva. El desarrollo de software no es, sin duda alguna, como la manufactura. Ambos dominios exhiben atributos completamente diferentes. La industria manufacturera tiene baja variabilidad, mientras que gran parte del desarrollo de software es altamente variable e intenta explotar la variabilidad a través de la novedad del diseño con el fin de generar ganancias. El software es por naturaleza “suave” y puede ser cambiado de manera fácil y barata, mientras que la manufactura tiende a ser sobre cosas “duras” que son difíciles de cambiar. Es natural ser escépticos sobre el valor de los sistemas Kanban en el desarrollo de software y otros trabajos de TI. Gran parte de lo que nosotros, como comunidad, hemos aprendido acerca de Kanban en los últimos años es contra-intuitivo. Nadie predijo el efecto en la cultura o la mejora de la colaboración entre funciones visto en Corbis y descrito en el capítulo 5. En estas páginas espero mostrarle que “¡Kanban puede!” Al hacerlo, espero poder convencerlo de que las reglas simples de Kanban pueden mejorar la productividad, reducir los plazos de entrega, mejorar la predicción, mejorar la satisfacción del cliente y con todo ello la cultura de su organización cambiará aumentando el trabajo colaborativo y las relaciones de trabajo funcional mejorará y se establecerán a través de su organización.

Para llevar

- ❖ Los sistemas kanban pertenecen a una familia de enfoques conocidos como sistemas de arrastre.
- ❖ La aplicación Drum-Buffer-Rope de la Teoría de Restricciones de Eliyahu Goldratt es una implementación alternativa de un sistema de arrastre.
- ❖ La motivación para adoptar un enfoque de sistema de arrastre es doble: el deseo de encontrar una forma sistemática para lograr un ritmo sustentable de trabajo y el deseo de encontrar un enfoque para introducir cambios en el proceso con aceptación incremental.
- ❖ Kanban es el mecanismo que se basa en el Sistema de Producción Toyota y su enfoque kaizen de mejora continua.
- ❖ El primer sistema kanban virtual para ingeniería de software fue implementado de 2004 al 2005 de Microsoft.
- ❖ Los resultados de las implementaciones iniciales de Kanban fueron muy alentadores en cuanto a la obtención de ritmo sustentable, la reducción de la resistencia al cambio a través del enfoque evolutivo incremental y la producción de mejoras económicas significativas.
- ❖ El Método Kanban como un enfoque para el cambio empezó a crecer con la adopción en la comunidad después de la conferencia Agile 2007 en Washington D.C. en agosto de 2007.
- ❖ A lo largo de este texto “kanban” (“k” minúscula) se utiliza para referirse a las tarjetas de señal y “sistema kanban” (“k” minúscula) se refiere a un sistema de arrastre implementado con tarjetas (virtuales) de señal.
- ❖ Kanban (“K” mayúscula) se utiliza para referirse a la metodología de mejora de procesos incremental evolutivo que surgió en Corbis entre 2006 y 2008 y ha seguido evolucionando en la comunidad más amplia de desarrollo de software Lean en los años transcurridos desde entonces.

❖ CAPÍTULO 2 ❖

¿Qué es el método Kanban?

En la primavera de 2005, durante la temporada de florecimiento de los cerezos a principios de abril, tuve la fortuna de tomar unas vacaciones en Tokio. Para disfrutar de este espectáculo hice mi segunda visita a los jardines en el lado Este del Palacio Imperial en el centro de Tokio. Fue allí donde tuve la revelación de que kanban no es tan sólo para manufactura.

El sábado 9 de abril de 2005, llegué al parque por la entrada Norte, cruzando el puente sobre el foso cerca de la estación de metro *Takebashi*. Muchos habitantes de Tokio estaban aprovechando la mañana del sábado soleado para disfrutar de la tranquilidad del parque y la belleza de los *sakura* (flor del cerezo).

La costumbre de tomar un picnic bajo los cerezos mientras que las flores caen a su alrededor se le conoce como *hanami* (fiesta de las flores). Es una tradición antigua en Japón. Es una oportunidad para reflexionar sobre la belleza, fragilidad y brevedad de la vida. La breve vida de la flor del cerezo es una metáfora de nuestra propia vida y de nuestra existencia corta, bella y frágil en la inmensidad del universo.

La flor del cerezo proporciona un contraste frente a los edificios grises del centro de Tokio, su bullicio, su palpitante multitud de gente ocupada y el ruido del tráfico. Los jardines son un oasis de tranquilidad y belleza en el corazón de la jungla de cemento. Conforme crucé el puente con mi familia, un hombre japonés de edad avanzada, con una maleta al hombro se acercó a nosotros y de su bolsa sacó un puñado de tarjetas de

plástico. Nos ofreció una tarjeta a cada uno haciendo una pausa breve para decidir si mi hija de 3 meses de edad, atada a mi pecho, requería de una tarjeta. El decidió que sí y me entregó dos tarjetas. No nos dijo nada y como mi japonés es limitado no inicié ninguna conversación. Seguimos caminando y entramos a los jardines para buscar un lugar donde disfrutar de nuestro picnic familiar.

Dos horas más tarde, después de una mañana agradable bajo el sol, empacamos la manta de picnic, recogimos la basura de nuestros bocadillos y nos dirigimos hacia la salida. Esta vez nos dirigimos a la Puerta Oriente en *Otemachi*. Cuando nos acercamos a la salida nos unimos a una fila de personas formadas frente a un pequeño quiosco. Conforme la línea avanzó vi que las personas estaban devolviendo sus tarjetas de plástico. Busqué en mi bolso y saqué las dos tarjetas que el hombre a la entrada me había dado. Al acercarme, vi dentro del quiosco a una señora japonesa pulcramente uniformada. Entre nosotros había un vidrio con un agujero semicircular cortado al nivel del mostrador, muy similar a las taquillas de admisión en un cinema o un parque de diversiones. Deslicé mis tarjetas de plástico por el mostrador a través del agujero en el vidrio. La señora las tomó con sus manos enguantadas en blanco y las apiló en un estante con las demás tarjetas. Hizo una ligera reverencia y me dio las gracias con una sonrisa. No hubo intercambio de dinero. No se ofreció explicación alguna del por qué yo había tenido que cargar 2 tarjetas blancas de admisión de plástico desde que entramos al parque 2 horas antes.

¿Qué estaba sucediendo en cuanto a las tarjetas de admisión? ¿Por qué molestarse en emitir un ticket si ningún dinero era colectado? Mi primer impulso fue que se trataba de un esquema de seguridad. Contando todas las cartas devueltas, las autoridades se asegurarían de que ningún visitante ambulante se hubiera perdido dentro de los terrenos a la hora de cerrar el parque en la tarde. Sin embargo, reflexioné al respecto y rápidamente me di cuenta de que sería un sistema de seguridad muy pobre. ¿Quién podría decir que me habían emitido dos tarjetas en lugar de tan solo una? ¿Contaba mi hija de 3 meses de edad como equipaje o como visitante? Parecía haber demasiada variabilidad en el sistema. Demasiadas oportunidades de error! Si se trataba de un esquema de seguridad entonces seguramente fracasaría y produciría falsos positivos todos los días. (Nota breve, tal sistema no puede producir falsos negativos puesto que requeriría de la fabricación de tarjetas adicionales. Este es un atributo útil común en los sistemas kanban.) Mientras tanto, las tropas estarían fuera corriendo alrededor de los arbustos todas las noches en busca de turistas perdidos. No, tenía que tratarse de otra cosa. Me di cuenta entonces de que los jardines del Palacio Imperial estaba usando un sistema kanban!



Photos this page courtesy of Thomas Blomseth

Esta epifanía fue muy esclarecedora y me dio permiso para pensar sobre los sistemas kanban más allá de la manufactura. Parecía probable que las fichas Kanban sean útiles en todo tipo de situaciones de gestión.

¿Qué es un sistema kanban?

Un número de kanban (o tarjetas) equivalente a la capacidad (acordada) del sistema es puesto en circulación. Una tarjeta se adjuntada a una parte del trabajo. Cada tarjeta actúa como un mecanismo de señalización. Una nueva parte del trabajo puede iniciarse solamente cuando una tarjeta está disponible. Esta tarjeta libre es adjuntada a una parte del trabajo y se permanece con ese trabajo a medida que fluye a través del sistema. Cuando no hay más tarjetas libres, no se puede iniciar ningún trabajo adicional. Todo trabajo nuevo tiene que esperar en cola hasta que una tarjeta esté disponible. Cuando algún trabajo ha sido completado, su tarjeta se desprende y se recicla. Ahora se puede iniciar una nueva parte del trabajo en cola con una tarjeta libre.

A este mecanismo se le conoce como un sistema de arrastre porque el nuevo trabajo se introduce en el sistema solamente cuando hay capacidad para llevarlo a cabo, en vez de ser empujado en el sistema a partir de la demanda. Un sistema de arrastre no se puede sobrecargar si la capacidad se ha establecido adecuadamente y está determinada por el número de tarjetas en circulación.

En el caso de los jardines del Palacio Imperial, el sistema son los jardines mismos, el trabajo-en-progreso son los visitantes y la capacidad está limitada por el número de tarjetas de admisión en circulación. Los visitantes recién llegados son admitidos únicamente cuando hay tarjetas disponibles para darles. En un día normal esto no es un problema. Sin embargo, en días de gran afluencia tales como días festivos y los sábados durante la temporada de la flor de cerezo, el parque es muy popular. Cuando se han entregado todas las tarjetas de admisión los visitante nuevos deben hacer cola a lo largo del puente y esperar las tarjetas recicladas conforme otros visitantes se van. El sistema kanban proporciona un método sencillo, barato y de fácil implementación para controlar el tamaño de la multitud limitando el número de personas dentro del parque. Esto les permite a los guardabosques mantener los jardines en buen estado y evitar daños ocasionados por demasiado tráfico y por atestamiento.

Kanban aplicado en el desarrollo de software

En el desarrollo de software, estamos usando un sistema virtual kanban para limitar el trabajo-en-progreso. Mientras que “kanban” significa “tarjeta de señal” y hay tarjetas utilizadas en la mayoría de las implementaciones de Kanban en desarrollo de software, estas tarjetas en realidad no funcionan como señales para iniciar más trabajo. En cambio, representan

ítems de trabajo. De ahí el término “virtual” porque no hay una tarjeta de señal física. La señal para iniciar un nuevo trabajo se deduce a partir de la cantidad visual de trabajo-en-progreso restado de algún indicador de límite (o capacidad). Algunos practicantes han implementado kanban físico utilizando técnicas tales como clips adhesivos o usando ranuras físicas en un tablero. Más a menudo la “señal” se genera a partir de un sistema de software para seguimiento de trabajo. Ejemplos de toda la mecánica de sistemas kanban aplicadas a trabajo de TI se encuentran en el Capítulo 6.

Las paredes de tarjetas se han convertido en un mecanismo de control visual popular en el desarrollo ágil de software, como se muestra en la Figura 2.1. El uso de un tablero de anuncios de corcho con tarjetas clavadas en ella o de un pizarrón blanco con notas adhesivas para seguimiento del trabajo-en-progreso se han convertido en la práctica común. Vale la pena observar en esta primera etapa que a pesar de algunos comentarios de la comunidad sobre lo contrario, estas paredes de tarjetas no son inherentemente sistemas kanban. Son tan solo sistemas de control visual. Le permiten a los equipos observar visualmente el trabajo-en-progreso; y a los equipos mismos auto-organizarse, asignar sus propias tareas y mover el trabajo del *backlog* para completarlo sin la dirección de un director de proyecto o línea. Sin embargo, si no hay un límite explícito del trabajo-en-progreso y no hay señal para tomar un nuevo trabajo a través del sistema, entonces no es un sistema kanban. Esto está explicado ampliamente en el capítulo 7.



Figura 2.1 Una pared de tarjetas de kanban (Cortesía de SEP)

¿Por qué usar un sistema kanban?

Como debe hacerse evidente en los capítulos siguientes, utilizamos un sistema kanban para limitar la capacidad del trabajo-en-progreso en un equipo y para equilibrar la demanda en el equipo contra el throughput de su trabajo entregado. De esa manera podemos alcanzar un ritmo sustentable de desarrollo, de modo tal que todas las personas pueden lograr un equilibrio entre trabajo y vida personal. Como podrá ver, Kanban también hace rápidamente obvios los asuntos que afectan el desempeño y reta al equipo a enfocarse en resolver esos asuntos con el fin de mantener un flujo continuo de trabajo. Dando visibilidad a los problemas de calidad y de proceso se hace obvio el impacto de los defectos, los cuellos de botella, la variabilidad y los costos económicos en el flujo y en el throughput. El simple hecho de limitar el trabajo-en-progreso con kanban fomenta mayor calidad y mayor rendimiento. La combinación de mejor flujo y mejor calidad contribuye a reducir el tiempo de entrega y a mejorar tanto la previsibilidad como el rendimiento para cumplir con la fecha de vencimiento. Mediante el establecimiento de una cadencia regular de entrega y entregando consistentemente en base a ella, Kanban ayuda a generar confianza de parte de los clientes y también de otros departamentos, proveedores y asociados en niveles posteriores del proceso a lo largo de la cadena de valor.

Haciendo todo esto, Kanban contribuye a la evolución cultural de las organizaciones. Exponiendo los problemas, enfocando la organización en resolverlos y eliminando sus efectos en el futuro, Kanban facilita el surgimiento de una organización de mejora continua muy capacitada que es altamente colaborativa y tiene mucha confianza.

Kanban ha demostrado mejorar la satisfacción del cliente a través de entregas frecuentes de software valioso, confiable y de alta calidad. Ha demostrado mejorar la productividad, la calidad y los plazos de entrega. Ha demostrado ser un mecanismo esencial que cataliza el surgimiento de una organización más ágil a través de un cambio cultural evolutivo.

El resto de este libro está dedicado a ayudarle a entender el uso de sistemas kanban en el desarrollo de software y a enseñarle cómo implementar tal sistema para lograr estos beneficios con su equipo.

Kanban, un Sistema Adaptivo Complejo para Lean

El método Kanban introduce un sistema adaptivo complejo que tiene la intención de catalizar un resultado Lean dentro de una organización. Los sistemas adaptivos complejos tienen condiciones iniciales y reglas simples requeridas para generar comportamiento complejo, adaptivo, y emergente. Kanban utiliza cinco propiedades centrales para crear un conjunto de comportamientos Lean en las organizaciones. Estas propiedades han estado presentes en toda implementación exitosa de kanban, incluyendo aquella en Microsoft descrita en el capítulo 4. Las cinco propiedades son:

1. Visualizar el Flujo de Trabajo
2. Limitar el Trabajo-en-Progreso
3. Medir y Administrar el Flujo
4. Hacer las Políticas del Proceso Explícitas
5. Utilizar Modelos* para Reconocer Oportunidades de Mejora

Hay al menos cinco propiedades emergentes que deberíamos esperar en una implementación de Kanban. Todas ellas surgieron en Corbis y por lo menos algunas de ellas han surgido en la mayoría de las implementaciones presentadas en conferencias y han sido documentadas dentro de la comunidad en los últimos dos años.

Nota: Ésta es una elección de política. Una solicitud de cambio por desarrollador en un momento dado es una política. Puede modificarse más tarde. Pensar sobre un proceso como un conjunto de políticas es un elemento clave del método Kanban.

1. Priorizar el Trabajo por Costo (Oportunidad) de Retrazo
2. Optimizar Valor por medio de Clases de Servicio
3. Distribuir el Riesgo con la Asignación de Capacidad
4. Incentivar la Innovación de Procesos
5. Administrar Cuantitativamente

Comportamiento surgidos con Kanban

Hay una lista creciente de comportamientos que están surgiendo, y hemos llegado a esperar, en las implementaciones de Kanban. Algunos o todos ellos han aparecido en la mayoría de las implementaciones recientes; todos ellos surgieron en Corbis durante el año 2007. Esperamos que esta lista crezca conforme aprendemos más sobre los beneficios de Kanban en las organizaciones.

1. Procesos personalizados por cada proyecto o flujo de valor
2. Cadencias desacopladas (o desarrollo sin iteraciones)

* Modelos comunes usados con Kanban incluyen la Teoría de Restricciones, Pensamiento en Sistemas, algún entendimiento de variación mediante las enseñanzas de W. Edwards Deming, y el concepto de *muda* (desperdicio) del Sistema de Producción Toyota. Los modelos usados en Kanban están evolucionando continuamente, e ideas de otros campos tales como psicología, sociología y gestión de riesgos están apareciendo en algunas implementaciones.

3. Trabajo programado en base a la oportunidad o Costo de Retraso
4. Valor optimizado mediante Clases de Servicio
5. Manejo de riesgos mediante la Asignación de Capacidad
6. Tolerancia para Experimentación sobre el Proceso
7. Gestión Cuantitativa
8. Difusión viral (de Kanban) a través de la Organización
9. Equipos pequeños fusionados para labor más fluida

Kanban como un otorgador de permisos

Kanban no es una metodología para el ciclo de vida del desarrollo de software o un enfoque para la gestión de proyectos. Se requiere que algún proceso esté ya establecido para que Kanban pueda aplicarse con el fin de cambiar gradualmente el proceso subyacente.

Este enfoque evolutivo, promovedor de cambio incremental ha sido controvertido en la comunidad del desarrollo ágil de software debido a que sugiere que los equipos no deben adoptar un método definido o plantilla de proceso. Una industria de servicios y herramientas se ha desarrollado en torno a un pequeño conjunto de prácticas definidas en dos métodos populares de desarrollo ágil. Ahora bien, con Kanban los individuos y los equipos pueden tener la facultad de desarrollar sus propias soluciones de proceso único, eliminando la necesidad de dichos servicios y requiriendo un nuevo conjunto de herramientas. De hecho, Kanban ha alentado una nueva ola de proveedores insurgentes de herramientas dispuestos a desplazar las herramientas ágiles actuales de gestión de proyectos con algo más visual y programable que puede ser fácilmente adaptado a un flujo de trabajo específico.

En los días iniciales del desarrollo de software ágil los líderes de la comunidad a menudo no entendían por qué sus métodos funcionaban. Hablábamos de “ecosistemas”¹⁰ y recomendábamos que los ejecutores siguieran todas las prácticas o la solución probablemente fallaría. Durante los años recientes ha habido una tendencia negativa que extiende esta forma de pensar. Unas cuantas empresas han publicado Modelos de Madurez Ágil que tratan de evaluar su adopción práctica. La comunidad de Scrum tiene una prueba basada en práctica a menudo referida como la “Prueba Nokia”¹¹. Estas evaluaciones basadas en práctica están diseñadas para impulsar la conformidad y negar la necesidad de una adaptación basada en contexto. Kanban le está facilitando al mercado ignorar estos esquemas de evaluación basados en práctica. Kanban está promoviendo activamente la diversidad.

En 2007, varias personas visitaron mis oficinas de Corbis para ver Kanban en acción. La pregunta común de todo visitante relacionado con la comunidad del desarrollo ágil de software podría parafrasearse como: “David, hemos recorrido el edificio y hemos visto siete

tableros kanban; cada uno es diferente! Cada equipo está siguiendo un proceso diferente! ¿Cómo es posible lidiar con esta complejidad?" Mi respuesta siempre fue "¡Por supuesto! La situación de cada equipo es diferente. Evolucionan sus procesos para que se adapten a su contexto". Pero yo sabía que estos procesos se derivaban de los mismos principios y era debido a que los miembros del equipo entendían estos principios básicos que eran capaces de adaptarse al ser reasignados de un equipo a otro.

A medida que más gente ha probado Kanban se han dado cuenta de que les ayuda a resolver los problemas que han encontrado con la gestión de cambios en sus organizaciones. Kanban le ha permitido a su equipo, proyecto u organización tener mejor agilidad. Hemos llegado a reconocer que Kanban está otorgando permiso en el mercado para crear

un proceso a la medida optimizado para un contexto específico. Kanban le está otorgando permiso a la gente para pensar por sí mismos. Le está otorgando permiso a la gente para ser diferente: diferente del equipo al otro lado del piso del edificio, en el siguiente piso, en el edificio de al lado y en un negocio vecino. Le está otorgando permiso a la gente para apartarse de los libros de texto. Lo mejor de todo es que Kanban nos proporciona las herramientas que nos permiten explicar (y justificar) por qué es mejor ser diferente y por qué la opción de ser diferente es la opción correcta en ese contexto. Para enfatizar esta elección diseñé una camiseta para la *Limited WIP Society* inspirado por el póster de la campaña de Obama diseñada por Shepard Fairey y con la

cara de Taiichi Ohno, el creador del sistema kanban de Toyota. El lema "Yes We Kanban" está diseñado para enfatizar que tiene otorgado el permiso. Usted tiene permiso para probar Kanban. Usted tiene permiso para modificar su proceso. Usted tiene permiso para ser diferente. Su situación es única y usted se merece desarrollar una definición de proceso única, adaptada y optimizada para su dominio, su cadena de valor, los riesgos que administra, las habilidades de su equipo y las exigencias de sus clientes.



Para llevar

- ❖ Los sistemas kanban se pueden utilizar en cualquier situación en la que hay un deseo de limitar la cantidad de cosas dentro de un sistema.
- ❖ Los jardines del Palacio Imperial en Tokio utilizan un sistema kanban para controlar el tamaño de la multitud en el parque.
- ❖ La cantidad de tarjetas de “kanban” en circulación limitan el trabajo-en-progreso.
- ❖ El nuevo trabajo es arrastrado dentro del proceso por una tarjeta de señal devuelta al finalizar una orden de trabajo o la tarea actual.
- ❖ En el trabajo de TI estamos, en general, utilizando un sistema kanban virtual debido a que no se utiliza ninguna tarjeta física real para definir el límite del trabajo-en-progreso.
- ❖ Las paredes de tarjetas habituales en el desarrollo ágil de software no son sistemas kanban.
- ❖ Los sistemas kanban crean una tensión positiva en el lugar de trabajo tal que exige la discusión de problemas.
- ❖ El Método Kanban (con K mayúscula) utiliza un sistema kanban como un catalizador del cambio.
- ❖ Kanban requiere que las políticas de proceso estén definidas explícitamente.
- ❖ Kanban utiliza herramientas de diversas áreas de conocimiento para fomentar el análisis de problemas y el descubrimiento de soluciones.
- ❖ Kanban permite la mejora incremental de procesos a través del descubrimiento repetitivo de los problemas que afectan el rendimiento del proceso.
- ❖ Una definición contemporánea del Método Kanban pueden encontrarse en línea en el sitio web de la *Limited WIP Society*
<http://www.limitedwipsociety.org/>
- ❖ Kanban está actuando como un otorgador de permiso en la profesión de desarrollo de software, motivando a los equipos a diseñar soluciones de procesos específicos al contexto en lugar de seguir una definición o plantilla del ciclo de vida de desarrollo de software de manera dogmática.

❖ PARTE DOS ❖

BENEFICIOS DE KANBAN

❖ CAPÍTULO 3 ❖

Una receta para el éxito

Durante la última década he tenido el reto de responder la siguiente pregunta: Como director, ¿qué acciones debe usted tomar cuando hereda un equipo, especialmente uno que no está trabajando de manera ágil, que tiene un amplio rango de habilidades y es, tal vez, completamente disfuncional? Por lo general he sido puesto en roles de dirección como agente de cambio. Por lo que he sido retado a crear un cambio positivo y en hacer progreso rápidamente, dentro de dos o tres meses.

Como director en organizaciones más grandes, nunca he sido capaz de contratar a mi propio equipo. Siempre se me ha pedido que adopte a un equipo que ya existe y, con cambios mínimos de personal, que genere una revolución en el rendimiento de la organización. Creo que esta situación es mucho más común que aquella en la que uno llega a contratar a un equipo entero nuevo.

Gradualmente desarrollé una manera de gestionar tal cambio. Está basada en experiencia, la cual incluye aprender de los fracasos que resultaron de tratar de usar el poder jerárquico para imponer un proceso y un flujo de trabajo. El mandato jerárquico tendió a fallar. Cuando le pedía a los equipos que cambiaran su comportamiento y utilizaran un método ágil tal como el Desarrollo Basado en Características; en inglés *Feature-Driven Development—FDD*), encontré resistencia. Contaba con sugerirles que no debían temer porque yo les iba a dar el entrenamiento y coaching requerido. A lo más obtuve aquiescencia; pero en realidad más bien se trataba de una institucionalización profunda del cambio. Pedirle

a la gente que cambie su comportamiento genera temor y reduce la autoestima debido a que comunica que claramente sus habilidades ya no son valoradas .

He desarrollado lo que he llegado a llamar mi Receta para el éxito para abordar estas situaciones. La Receta para el éxito presenta las directrices para un nuevo director que adopta a un equipo que ya existe. Siguiendo la receta se obtiene una mejoría rápida con bajos niveles de resistencia por parte del equipo. Quiero reconocer aquí la influencia directa de Donald Reinertsen, quien me dio los dos primeros y los últimos pasos de la receta; así como la influencia indirecta de Eli Goldratt, cuyos escritos sobre la Teoría de Restricciones y sus Cinco Pasos de Enfoque influenciaron los pasos cuatro y cinco .

Los seis pasos de la receta son:

- Enfocarse en la Calidad
- Reducir el Trabajo-en-Progreso
- Entregar a Menudo
- Equilibrar la Demanda Contra el throughput
- Priorizar
- Atacar las Fuentes de Variabilidad para Mejorar la Previsibilidad

Implementando la receta

Un director técnico debe seguir la receta en orden. El enfoque en la Calidad es lo primero, ya que está bajo el exclusivo control e influencia de un director tal como un director de desarrollo o de pruebas o el supervisor del director, con un título tal como Director de Ingeniería. Siguiendo la receta, el control se hace gradualmente menor, se requiere una mayor colaboración con los grupos en ambas direcciones del proceso hasta llegar al paso de Priorización. Establecer prioridades es justamente el trabajo del sector de negocio, no de la organización de tecnología, por lo que no debería estar dentro del ámbito del director técnico. Por desgracia, normalmente el sector de negocio renuncia a priorizar el trabajo y pasa a un director técnico esa responsabilidad de dar prioridad al trabajo—y luego culpa a ese gestor por tomar malas decisiones. Atacar las fuentes de variabilidad para mejorar la previsibilidad esta al final de la lista porque algunos tipos de variabilidad requieren cambios de conducta con el fin de reducirlos. Pedirle a la gente que cambie su comportamiento es difícil! Por lo que atacar la variabilidad es mejor cuando hay un resultado del cambio climático a partir de algunos éxitos con los pasos anteriores. Como veremos en el capítulo 4, a veces es necesario analizar las causas de la variabilidad con el fin de llevar a cabo algunos

de los pasos anteriores. El truco es escoger fuentes de variabilidad que requieran pocos cambios de comportamiento y que puedan ser fácilmente aceptadas.

Enfocarse en la calidad es más fácil porque es una disciplina técnica que puede ser dirigida por el director de la función. Los otros pasos tienen mayor reto porque dependen de un acuerdo y de la colaboración de otros equipos. Se requieren habilidades en articulación, negociación, psicología, sociología e inteligencia emocional. Llegar a un consenso en torno a la necesidad de equilibrar la Demanda contra el throughput es crucial. Resolver problemas relacionados con la disfunción entre los roles y las responsabilidades de los miembros del equipo requiere una mayor capacidad de negociación y diplomacia. Tiene sentido, entonces, ir tras las cosas que están directamente bajo su control y que usted sabe que tienen un efecto positivo en la ejecución tanto su equipo como de su negocio.

Desarrollando un mayor nivel de confianza con los otros equipos puede permitir tratar las cosas más difíciles. Construir y mostrar código de alta calidad con pocos defectos mejora la confianza. Entregar código de alta calidad con regularidad genera aún más confianza. Conforme el nivel de confianza aumenta, el director obtiene mayor valor político. Esto permite pasar al siguiente paso de la receta. Al final su equipo ganará el respeto suficiente para que usted sea capaz de influenciar a los propietarios del producto, el equipo de marketing y los patrocinadores empresariales para cambiar su comportamiento y podrán colaborar para dar prioridad al trabajo más valioso que debe desarrollarse.

Es difícil atacar las fuentes de variabilidad para mejorar la previsibilidad. Esa tarea no debe llevarse a cabo sino hasta que el equipo esté actuando a un nivel más maduro y haya mejorado mucho. Los primeros cuatro pasos de la receta tendrán un impacto significativo. Ellos le darán el éxito al nuevo director. Sin embargo, para crear realmente una cultura de innovación y mejora continua, será necesario atacar las fuentes de variabilidad en el proceso. Por lo que el paso final en la receta es a modo de crédito extra: Es el paso que separa a los grandes líderes técnicos de aquellos meramente competentes.

Enfoque en calidad

El Manifiesto Ágil no dice nada acerca de la calidad, aunque los Principios detrás del Manifesto¹² hablan sobre artesanía y hay un enfoque implícito en la calidad. Entonces, si la calidad no aparece en el Manifiesto, ¿por qué es lo primero en mi receta para el éxito? En pocas palabras, los defectos excesivos son el mayor desperdicio en desarrollo de software. Los números a ese respecto son alarmantes y muestran varias órdenes de magnitud de la variación. Capers Jones¹³ informa que en el año 2000, durante la burbuja de las punto-com, midió que la calidad del software oscilaba entre 6 defectos por cada punto función a menos de 3 por cada 100 puntos de función, un rango de 200 a 1 para equipos en Norte-América. El punto medio es de aproximadamente 1 defecto por cada 0.6 a 1.0 puntos de función. Esto implica que es común que los equipos gasten más del 90 por ciento de su esfuerzo

corrigiendo defectos. Como evidencia directa, a finales de 2007, Aaron Sanders, un proponente temprano de Kanban, informó, en el grupo Yahoo! Kanbandev, que un equipo con el que él estaba trabajando dedicaba el 90 por ciento de su capacidad disponible en corregir defectos.

Fomentar la alta calidad desde el principio tiene un gran impacto en la productividad y el throughput de los equipos con altas tasas de defectos. Una mejora del rendimiento de dos a cuatro veces es razonable. Con equipos realmente malos, una mejora de diez veces puede lograrse si se enfocan en la calidad.

La mejora de la calidad del software es un problema bien entendido.

Tanto el enfoque de calidad para desarrollo ágil como los enfoques de calidad tradicionales tienen mérito. Deben utilizarse combinados. Ingenieros de pruebas profesionales deben efectuar las pruebas. El uso de ingenieros de pruebas nos permite encontrar defectos en el código, evitando que lleguen a producción. Pedirle a los desarrolladores que escriban pruebas unitarias y las automaticen para proporcionar pruebas de regresión automatizadas también tiene un efecto dramático. Parece haber una ventaja psicológica al pedirle a los desarrolladores que escriban las pruebas primero. Conocido como Desarrollo Basado en Pruebas; (o en inglés *Test-Driven Development* –TDD) parece proporcionar la ventaja de que las pruebas de cobertura son más completas. Vale la pena señalar que he dirigido equipos bien disciplinados que escribían las pruebas después de escribir el código funcional y disponían de liderazgo en calidad de liderazgo en la industria. Sin embargo, parece claro que, para el equipo promedio, insistir en escribir las pruebas primero, antes del código funcional, mejora la calidad.

Inspecciones de Código mejoran la calidad. Ya sea mediante programación en parejas, revisión por colega, code walkthrough, o inspecciones Fagan completas, las inspecciones de código funcionan. Ayudan a mejorar tanto la calidad externa como la calidad interna del código. Las inspecciones de código son mejores si se hacen con frecuencia y en pequeñas cantidades. Motivo a los equipos a inspeccionar el código por lo menos 30 minutos diarios.

El análisis y el diseño en colaboración mejoran la calidad. La calidad se incrementa cuando se les pide a los equipos que trabajen juntos para analizar problemas y soluciones de diseño. Motivo a los equipos a que lleven a cabo sesiones de equipos colaborativos para análisis y para modelado de diseño. El modelado de diseño se debe hacer en pequeñas cantidades todos los días. Scott Ambler lo llama Modelado Ágil¹⁴.

El uso de patrones de diseño mejora la calidad. Los patrones de diseño capturan soluciones conocidas a problemas conocidos. Los patrones de diseño aseguran que esté disponible más información de manera temprana en el ciclo de vida y que los defectos de diseño sean eliminados.

El uso de herramientas modernas de desarrollo mejora la calidad. Muchas de las herramientas modernas incluyen funciones para realizar un análisis estático y dinámico de código. Esto debe estar activado y sintonizado para cada proyecto. Estas herramientas

de análisis pueden evitar que los programadores cometan errores elementales tales como la introducción de problemas bien conocidos, por ejemplo, defectos de seguridad.

Herramientas modernas de desarrollo más exóticas tales como Líneas de Productos Software (o Fábricas de Software) y Lenguajes de Dominio Específico reducen los defectos. Las fábricas de Software puede usarse para encapsular patrones de diseño en forma de fragmentos de código, lo que reduce la potencial inserción de defectos al generar código. También se pueden utilizar para automatizar tareas repetitivas de codificación, reduciendo de nuevo el potencial de inserción de defectos al generar código. El uso de las fábricas de software también reduce la demanda por inspecciones de código, debido a que el código de fábrica no tiene necesidad de ser re-inspeccionado. Tiene una calidad conocida.

Algunas de estas últimas sugerencias pertenecen en realidad a la categoría de reducción de variabilidad en el proceso. El uso de las fábricas de software e incluso los patrones de diseño invitan a los desarrolladores a cambiar su comportamiento. La mayor ganancia viene de usar ingenieros de pruebas profesionales, escribir pruebas primero, automatizar las pruebas de regresión, e inspecciones de código. Y una cosa más...

La reducción de la cantidad de diseño-en-progreso impulsa la calidad del software.

Reduzca el Trabajo-en-Progreso y entregue seguido

En 2004 trabajé con dos equipos en Motorola. Ambos equipos estaban desarrollando el código del servidor de red para aplicaciones de telefonía celular. Un equipo estaba trabajando en el servidor de descarga por-el-aire (PA) para tonos, juegos, otras aplicaciones y datos. El otro equipo estaba desarrollando el servidor para la administración de dispositivos por-el-aire (AD PA). Ambos equipos estaban usando la metodología de FDD. Ambos equipos eran aproximadamente del mismo tamaño-alrededor de ocho desarrolladores, un arquitecto, un máximo de cinco ingenieros de pruebas y un director del proyecto. Trabajando en conjunto con la gente de mercadotecnia, los equipos fueron responsables de su propio análisis y diseño. Además, se contaba con equipos de diseño de experiencia del usuario y de documentación para usuario (escritos técnicos) los cuales prestaron servicio a ambos equipos del proyecto.

WIP, tiempo de entrega y Defectos

La figura 3.1 muestra un diagrama de flujo acumulado. En inglés:*Cummulative Flow Diagram—CFD*) sobre el trabajo del proyecto hecho por el equipo de descarga PA. Un diagrama de flujo acumulativo es una gráfica de área que representa la cantidad de trabajo en un estado determinado. Los estados que se muestran en esta gráfica son: “inventario”, que significa el *backlog* o la cola que aún no se ha iniciado; “comenzado”, el cual implica que los requisitos para una característica han sido explicados a los desarrolladores; “diseñado”,

que significa concretamente que un diagrama UML de secuencia se ha desarrollado para la característica; “codificado”, que significa que los métodos en el diagrama de secuencia se han realizado; y “completado”, lo que significa que todas las pruebas unitarias están pasando, que el código ha sido revisado y que los desarrolladores líderes de equipo han aceptado el código y lo han promovido para pruebas.

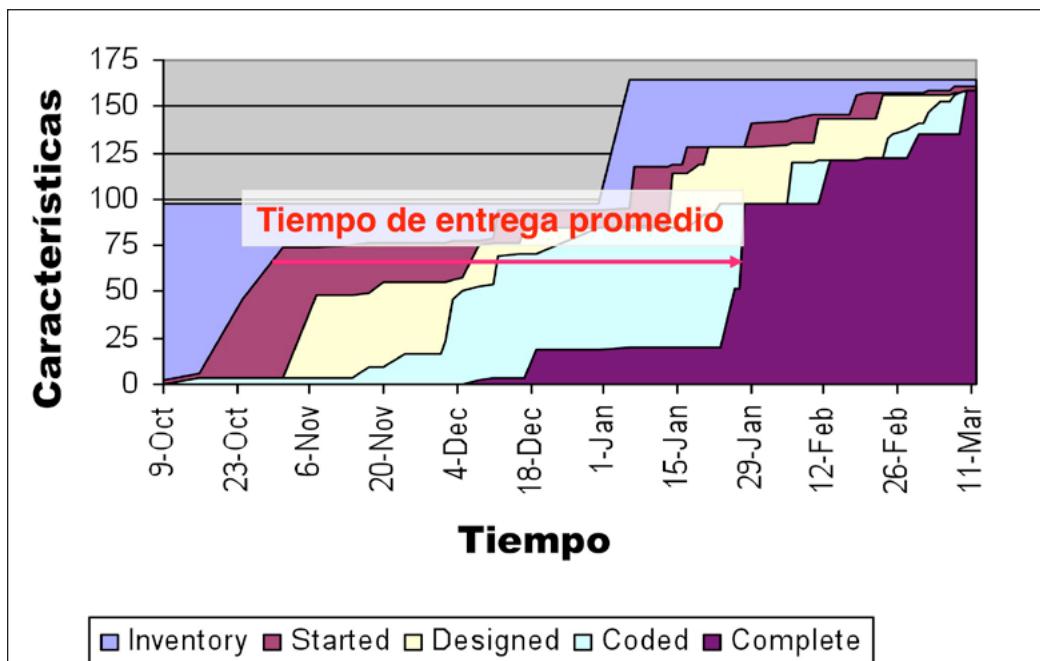


Figura 3.1 Diagrama de flujo acumulativo: equipo de descarga PA de otoño de 2003 al invierno de 2004

La primera línea en la gráfica muestra el número de características en el ámbito del proyecto. El alcance llegó en dos lotes de parte de los propietarios de negocios. La segunda línea muestra el número de características comenzadas. La tercera línea muestra el número de características diseñadas. La cuarta línea muestra parte del número de características desarrolladas y la quinta línea muestra el monto de características finalizadas y listas para pruebas.

La altura entre la segunda y quinta líneas en cualquier día muestra la cantidad de trabajo-en-progreso, mientras que la distancia horizontal entre la segunda y el quinta líneas muestra el tiempo de entrega promedio desde el día de inicio de una característica hasta que es terminada. Es importante tener en cuenta que la distancia horizontal es el tiempo de entrega promedio y no el tiempo de entrega específico para una característica específica. El diagrama de flujo acumulativo no hace seguimiento de características específicas. La característica número cincuenta y cinco puede que sea la característica treinta en ser terminada. No existe asociación entre una línea en el eje-Y y una característica específica en el *backlog*.

El equipo del servidor de descarga PA carecía de la disciplina o quizá carecía de convencimiento para utilizar el método de FDD. No trabajaban en colaboración, como lo exige el FDD. Repartían grandes cantidades de características a cada desarrollador. Por lo general, tenían diez características simultáneas en curso por desarrollador en cualquier momento dado. El equipo de AD PA siguió el método como si fuera definición de libro de texto. Estaban funcionando bien en colaboración. Desarrollaron las pruebas unitarias para el 100% de la funcionalidad. Y lo más importante es que estaban trabajando en pequeños lotes de características habitualmente entre cinco y diez características en progreso para todo el equipo en un momento dado. Como punto de referencia, una característica en FDD parece representar aproximadamente 1.6 a 2.0 puntos de función de código.

El equipo del servidor de descarga PA tuvo un tiempo de entrega promedio de alrededor de tres meses por unidad desde su inicio hasta completarla para mandarla de Seattle, Washington, a Champaign, Illinois, para efectuar pruebas de integración, como se muestra en la Figura 3.1. El equipo de AD PA, tenía un tiempo de entrega promedio por unidad en el rango de cinco a diez días, ilustrado en la Figura 3.2. La diferencia entre la calidad inicial, medida como el número de defectos fugados en el sistema o pruebas de integración, fue mayor de 30 veces entre los dos equipos. El equipo de AD PA produjo una calidad inicial a nivel del líder en la industria de dos o tres defectos por cada 100 características, mientras que el equipo del servidor de descarga PA produjo resultados a nivel del promedio en la industria de alrededor de dos defectos por característica.

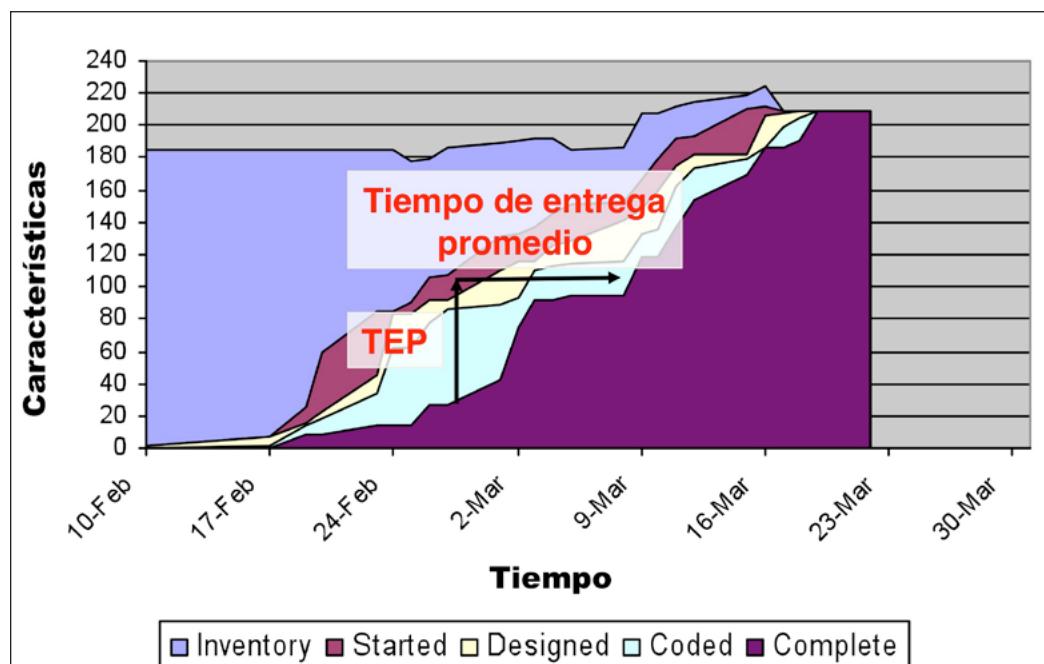


Figura 3.2 Diagrama de flujo acumulativo del equipo AD PA, invierno de 2004

Examinando los diagramas podemos ver que la cantidad de Trabajo-en-Progreso esta directamente relacionada con el tiempo de entrega. La Figura 3.2 muestra claramente que el tiempo promedio de espera disminuye a medida que la cantidad de trabajo-en-progreso baja. En el pico, el tiempo de entrega promedio es de doce días. Más adelante en el proyecto, con menos trabajo-en-progreso, el tiempo de entrega promedio bajó a tan sólo cuatro días.

Hay causalidad entre la cantidad de trabajo-en-progreso y el tiempo promedio de entrega y la relación es lineal. En la industria manufacturera se le conoce a esta relación como Ley de Little. La evidencia de estos dos equipos en Motorola sugiere que existe una correlación entre mayor tiempo de entrega y menor calidad. Tiempos de entrega largos parecen estar asociados con calidad significativamente más pobre. De hecho, un incremento promedio de aproximadamente seis y medio veces en el tiempo de espera resulta en un aumento de más de 30 veces en defectos iniciales. Mayores tiempos de entrega resultan a partir de mayores montos de trabajo-en-progreso. Por lo tanto, el punto de influencia en la gestión para mejorar la calidad es reducir la cantidad de trabajo-en-progreso. Desde el descubrimiento de esta evidencia he gestionado el trabajo-en-progreso como un medio de control de calidad y me he convencido de la relación entre la cantidad de WIP y la calidad inicial. Sin embargo, hasta el tiempo de escribir este libro no hay evidencia científica para respaldar este resultado empíricamente observado.

La reducción de trabajo-en-progreso, o la reducción de la duración de una iteración, tendrá un impacto significativo en la calidad inicial. Parece ser que la relación entre la cantidad de trabajo-en-progreso y la calidad inicial no es lineal; es decir, los defectos se incrementan de manera desproporcionada a los aumentos en la cantidad de WIP. Por lo que tiene sentido que las iteraciones de dos semanas son mejores que las iteraciones de cuatro semanas y que iteraciones de una semana son mejores aún. Iteraciones cortas impulsarán una mayor calidad.

Siguiendo la lógica de la evidencia presentada, tiene todavía más sentido simplemente limitar el trabajo-en-progreso utilizando un sistema kanban. Si sabemos que gestionando el WIP mejorará la calidad, ¿por qué no introducir una política explícita de limitar el WIP, liberando así a los directores para centrarse en otras actividades?

Debido a la estrecha interacción entre el trabajo-en-progreso y la calidad, se deduce que el paso 2 de la receta debe aplicarse junto con o poco después del paso 1.

¿Quién es mejor?

Intervine con el equipo de descarga PA durante la semana de la Navidad de 2003 y le sugerí al líder del equipo que había demasiado trabajo-en-progreso, que el tiempo de entrega era demasiado largo y que muy pocas cosas estaban siendo terminadas. Compartí mi preocupación de que esto provocaría una mala calidad. Él tomó de buen corazón mi consejo y en enero de 2004 introdujo algunos cambios en la forma en que el equipo trabajaba. El

resultado fue una reducción del WIP en 2004 y un tiempo de entrega claramente corto. Este cambio, sin embargo, llegó demasiado tarde para evitar que el equipo creara una gran cantidad de defectos.

Mientras que el diagrama sugiere que el proyecto fue completado a mediados de marzo de 2004, el equipo siguió trabajando en el software hasta mediados de julio de ese año. La mitad del equipo de AD PA fue retirado de su proyecto para ayudar a reparar los defectos. En julio de 2004 el director general de la unidad de negocios declaró el producto como completado a pesar de las continuas preocupaciones sobre su calidad. El producto le fue entregado al equipo de ingeniería de campo. Durante su despliegue, hasta un 50 por ciento de los clientes cancelaron la entrega debido a problemas de calidad. Mientras el equipo de ingeniería de campo mantuvo buenas relaciones personales con el equipo de ingeniería del producto, tenían un bajo nivel de respeto profesional y carecían de confianza en la capacidad del equipo. Su opinión era que el producto era de mala calidad y el equipo era incapaz de ofrecer algo mejor.

Irónicamente, si en ese entonces usted hubiera entrado en ese edificio en el barrio-SoDo de Seattle en ese entonces y le hubiera preguntado a los desarrolladores, “¿Quién es la persona más inteligente por aquí?”, ellos le hubieran señalado a alguien del equipo de descarga PA. Si usted les preguntara: “¿Quién tiene más experiencia?” tendría de nuevo la misma respuesta. Si usted viera sus currículos vitae se daría cuenta de que la experiencia promedio del equipo de descarga PA es superior a la del equipo de AD PA en tres años. En papel todo sugería que el equipo de descarga PA era mejor. Y hasta hoy en día, algunos de estos individuos creen que ellos son mejores a pesar de todas las pruebas cuantitativas que indican lo contrario.

Por medio de mi experiencia como director y coach con este equipo supe que algunos de los miembros del equipo AD PA sufrían de baja autoestima profesional y se preocupaban de no ser tan talentosos como algunas de las personas verdaderamente inteligentes en el otro equipo. Sin embargo, la productividad del equipo de AD PA era cinco veces y media mayor y con más de treinta veces mejor calidad inicial. El proceso adecuado, con buena disciplina, buena gestión y buen liderazgo hicieron toda la diferencia. Lo que este ejemplo realmente demuestra es que usted no necesita de la mejor gente para producir resultados de calidad mundial. Una de las creencias básicas en la comunidad ágil, a lo que me refiero como el “esnobismo artesanal”, sugiere que todo lo que necesitas para tener éxito en el desarrollo ágil es un pequeño equipo de gente realmente buena. Sin embargo, en este caso, un equipo con amplitud de talento fue capaz de producir resultados de calidad mundial.

Entregas Frecuentes Generan Confianza

La reducción del WIP acorta el tiempo de entrega. Tiempos de entrega más cortos significan que es posible entregar más seguido código que funciona. Entregas más frecuentes

generan confianza por parte de los equipos externos, particularmente el equipo de marketing o los patrocinadores de negocio. Es difícil definir confianza. Los sociólogos lo llaman capital social. Lo que han aprendido es que la confianza está dirigida por eventos y que gestos o eventos pequeños y frecuentes aumentan la confianza más que grandes gestos ocasionales.

Cuando enseño esto al dar clases me gusta preguntarles a las mujeres en la clase lo que piensan después de tener una primera cita con un chico. Les sugiero que la pasaron muy bien pero entonces él no las llama durante dos semanas. A continuación, aparece él en su puerta con un ramo de flores y una disculpa. Les pido que comparan eso con alguien que se toma el tiempo para escribir un mensaje de texto en camino de su casa esa noche para decir, "Pasé un tiempo muy agradable esta noche. Realmente quiero volver a verte. ¿Te llamo mañana?" Y de hecho llama al día siguiente. ¿Adivina a quién prefieren? Pequeños gestos a menudo no cuestan nada pero construyen más confianza que gestos grandes, caros (o expansivos) otorgado de vez en cuando.

Así mismo ocurre con el desarrollo de software. Entregas pequeñas y frecuentes de alta calidad construyen más confianza con los equipos asociados que con entregas grandes y menos frecuentes.

Liberaciones pequeñas demuestran que el equipo de desarrollo de software puede entregar y está comprometido a proporcionar valor. Generan confianza con el equipo de marketing o de empresas patrocinadoras. Alta calidad en el código entregado gana la confianza de los socios tales como operaciones, soporte técnico, ingeniería de campo y ventas.

Conocimiento tácito

Es muy fácil especular acerca del por qué pequeños lotes de código mejoran la calidad. La complejidad de los problemas del trabajo en el conocimiento crece exponencialmente con la cantidad de trabajo-en-progreso. Mientras tanto nuestros cerebros humanos luchan para hacer frente a toda esta complejidad. Gran parte de la transferencia de conocimiento y descubrimiento de información en el desarrollo de software es de naturaleza tácita y se crea durante sesiones de trabajo en colaboración, cara-a-cara. La información es verbal y visual, pero tiene un formato informal, tal como un dibujo en un tablero. Nuestras mentes tienen una capacidad limitada para almacenar todo ese conocimiento tácito y que se degrada mientras la almacenamos. No recordamos detalles precisos y cometemos errores. Si un equipo está co-localizado y siempre disponible entre sus miembros, esta pérdida de memoria se puede corregir mediante el debate frecuente o tocando la memoria compartida de un grupo de personas. Así que los equipos ágiles co-localizados en un espacio de trabajo compartido son más propensos a retener el conocimiento tácito por más tiempo. Independientemente de esto, el conocimiento tácito se deprecia con el tiempo, por lo que tiempos de entrega más cortos son esenciales para los procesos de conocimiento tácito.

Sabemos que la reducción de trabajo-en-proceso está directamente relacionado con la reducción de tiempo de entrega. Por lo tanto, podemos inferir que habrá menor pérdida de conocimiento tácito cuando tenemos menos trabajo-en-progreso, resultando en una mayor calidad.

En resumen, la reducción del trabajo-en-progreso mejora la calidad y permite entregas más frecuentes. Entregas más frecuentes de código de mayor calidad aumenta la confianza de los equipos externos.

Equilibrio de la demanda contra el throughput

Equilibrar la demanda en contra del throughput implica que vamos a establecer la tasa a la que aceptamos nuevos requerimientos en nuestra línea de desarrollo de software para que corresponda con la velocidad a la que podemos entregar código funcional. Cuando hacemos esto, estamos fijando de manera efectiva nuestro trabajo-en-progreso a un cierto tamaño. Conforme el trabajo es entregado, vamos a tomar un nuevo trabajo (o requerimiento) de las personas que crean la demanda. Así que cualquier discusión sobre las prioridades y el compromiso con un nuevo trabajo sólo puede ocurrir en el contexto de la entrega de trabajo que ya existe.

El efecto de este cambio es profundo. El throughput de su proceso se verá limitado por un cuello de botella. Es poco probable que usted sepa dónde se encuentra el cuello de botella. De hecho, si usted habla con todos los involucrados en la cadena de valor, probablemente todo afirmarán que están completamente sobrecargados. Sin embargo, una vez que equilibre la demanda contra el throughput y límite el trabajo-en-progreso dentro de su cadena de valor, se creará magia. Únicamente los recursos en el cuello de botella estarán completamente ocupados. De manera muy rápida otras personas en la cadena de valor se darán cuenta que tienen holgura . Mientras tanto, aquellos en el cuello de botella estarán ocupados, pero no sobrecargados. Por primera vez, probablemente en años, el equipo no estará sobrecargado y mucha gente tendrá una experiencia muy rara en sus carreras, la sensación de tener tiempo en sus manos.

Crear Holgura

Gran parte de la tensión será liberada de la organización y la gente será capaz de concentrarse en hacer su trabajo con precisión y calidad. Serán capaces de sentirse orgullosos de su trabajo y disfrutar de la experiencia aún más. Aquellos con tiempo en sus manos comenzarán a dedicar ese tiempo a mejorar sus propias circunstancias, podrán poner en orden su área de trabajo o recibir capacitación. Es probable que comiencen a dedicarse a mejorar sus habilidades, sus herramientas y la manera cómo interactúan con los demás en los niveles anteriores y posteriores del proceso. A medida que pase el tiempo y una pequeña mejora

lleva a otra, el equipo será visto como de mejora continua. La cultura habrá cambiado. La holgura creada por el hecho de limitar el trabajo-en-progreso y tomando nuevo trabajo sólo cuando hay capacidad disponible permitirá mejoras que nadie pensó como posible.

Usted necesita holgura para permitir la mejora continua. Necesita equilibrar la demanda contra el throughput y limitar el monto de trabajo-en-progreso para permitir holgura.

La gente cree intuitivamente que tienen que eliminar holgura. Por lo que, después de limitar el monto de trabajo-en-progreso mediante el equilibrio de la demanda contra el throughput, la tendencia es “equilibrar la línea” mediante ajustes de recursos para que todos estén completamente utilizados de manera eficiente. Aunque eso parezca eficiente y satisfaga las prácticas gerenciales típicas del siglo veinte, impedirá la creación de una cultura de mejora. Necesita holgura para permitir mejora continua. Para tener holgura, usted debe tener una cadena de valor desequilibrada con un recurso en cuello de botella. La optimización de la utilización no es deseable.

Priorizar

Si los primeros tres pasos de la receta han sido implementados las cosas estarán funcionando sin problemas. El código de alta calidad deberá llegar con frecuencia. Los tiempos de entrega deberán ser relativamente cortos conforme el trabajo-en-progreso es limitado. El nuevo trabajo será tomado y desarrollado únicamente conforme la capacidad es liberada al término de trabajo que ya existe. En este punto, la atención de la dirección puede tornarse a optimizar el valor entregado en lugar de a la cantidad de código entregado. No tiene mucho sentido el ponerle atención al establecimiento de prioridades cuando no hay previsibilidad en la entrega. ¿Por qué desperdiciar esfuerzo tratando de ordenar la entrada cuando no hay confiabilidad en el orden de la entrega? Mientras esto no se resuelva, el tiempo de gestión se utiliza mejor centrándose en mejorar tanto la capacidad de entregar como la previsibilidad de la entrega. Usted debe concentrar sus pensamientos en ordenar la prioridad de la entrada una vez que sabe que realmente puede entregar las cosas en aproximadamente el orden en que se solicitan.

Influencia

El establecimiento de prioridades no debe ser controlado por la organización de ingeniería y por lo tanto no está bajo el control de la dirección de ingeniería. La mejora de la priorización requiere que el propietario de producto, el patrocinador de negocio, o el departamento de marketing cambien su comportamiento. Cuando más, la dirección de ingeniería puede buscar influenciar la manera en que la priorización se efectúa.

Con el fin de tener capital social y político para influenciar el cambio, un cierto nivel de confianza debe haber sido establecido. Sin la capacidad de entregar código de alta calidad

con regularidad no puede haber confianza y por ende habrá poca posibilidad tanto de influir en la priorización así como de optimizar el valor entregado por el equipo de desarrollo.

Recientemente se ha vuelto popular entre la comunidad ágil el hablar sobre la optimización de valor para el negocio y como la tasa de producción de código funcional (conocido como “velocidad” del desarrollo de software) no es una métrica importante. Esto es porque el valor de negocio entregado es la verdadera medida de éxito. Mientras que esto puede ser a fin de cuentas cierto, es importante no perder de vista que la escala de madurez de la capacidad que un equipo debe ascender. La mayoría de las organizaciones son incapaces de medir y reportar el valor de negocio entregado. Primero deben construir la capacidad en las habilidades básicas antes de intentar retos más grandes.

La Construcción de la Madurez

Esta es la forma en que creo que un equipo debe madurar: Primero, aprender a construir código de alta calidad. Después reducir el monto de trabajo-en-progreso, acortar el tiempo de entrega y entregar seguido. A continuación, equilibrar la demanda contra el throughput, limitar el monto de trabajo-en-progreso y crear holgura para liberar disponibilidad, lo cual permitirá mejoras. Después, con capacidad de desarrollo de software funcionando y optimizando de manera suave, mejorar la priorización para optimizar la entrega de valor. Tener esperanza en la optimización del valor de negocio son tan solo deseos. Tome acciones para llegar a este nivel de madurez incrementalmente—siga la receta para el éxito.

Ataque las fuentes de variabilidad y mejore el proceso

Los efectos de la variabilidad y el cómo reducirlos dentro de un proceso son temas avanzados. La reducción de variabilidad en el desarrollo de software requiere que los trabajadores de conocimiento modifiquen la manera en que trabajan—para aprender nuevas técnicas y cambiar su comportamiento personal. Todo eso es difícil, por lo que no es para principiantes o para organizaciones inmaduras.

El resultado de la variabilidad es mayor trabajo-en-progreso y mayores tiempos de espera. Esto se explica de manera más completa en el capítulo 19. La variabilidad crea una necesidad mayor de holgura en recursos que no están en el cuello de botella con el fin de hacer frente al flujo y reflujo de trabajo conforme sus efectos se manifiestan a lo largo del flujo de trabajo. Un entendimiento completo sobre el porqué esto es cierto requiere de antecedentes en control estadístico de proceso y teoría de colas, lo cual está más allá del alcance de este libro. Personalmente, me gusta el trabajo sobre variabilidad y colas de Donald Wheeler y Donald Reinertsen; por lo que, si desea más información sobre esos temas, comience ahí.

Por ahora confíe en que la variabilidad en el tamaño de los requerimientos y en el monto de esfuerzo utilizado en análisis, diseño, codificación, pruebas, integración de construcciones de software y entregas afectan negativamente el throughput del proceso y los costos de ejecución de la cadena de valor de software desarrollado.

Aun así algunas fuentes de variabilidad son diseñadas dentro de los procesos mediante decisiones pobres sin darnos cuenta. El caso de estudio en el capítulo 4 muestra varios ejemplos: la replanificación mensual, el acuerdo de nivel de servicio en las estimaciones y la prioridad de los cambios de texto en producción. Estos tres ejemplos son controlados por políticas que se pueden cambiar. El cambio de la política de un proceso puede reducir dramáticamente las fuentes de variabilidad que afectan a la previsibilidad.

Receta para el éxito y Kanban

Kanban permite los seis pasos en la Receta Para el Éxito. Kanban cumple con la Receta Para el Éxito y la Receta Para el Éxito cumple con su promesa para el director que la implementa. A su vez, la Receta Para el Éxito ilustra por qué Kanban es una técnica tan valiosa.

Para llevar

- ❖ Kanban brinda todos los aspectos de la Receta Para el Éxito.
- ❖ La Receta Para el Éxito explica por qué Kanban tiene valor.
- ❖ La mala calidad puede representar el desperdicio más alto en el desarrollo de software.
- ❖ La reducción de trabajo-en-progreso mejora la calidad.
- ❖ La mejora de calidad mejora la confianza con los asociados en niveles posteriores del proceso tales como operaciones.
- ❖ Las entregas frecuentes mejoran la confianza con asociados en niveles anteriores del proceso tales como marketing.
- ❖ La demanda puede ser equilibrada enfrente al throughput con un sistema de arrastre.
- ❖ Los sistemas de arrastre exponen los cuellos de botella y generan holgura donde no hay cuellos de botella.
- ❖ La priorización de alta calidad maximiza el valor emitido por una cadena de valor de desarrollo de software que funciona bien.
- ❖ La priorización tiene poco valor sin buena calidad inicial y previsibilidad de entrega.
- ❖ El poder hacer cambios para reducir variabilidad requiere de holgura.
- ❖ La reducción de la variabilidad disminuye la necesidad de holgura.
- ❖ La reducción de la variabilidad permite el equilibrio de recursos (y potencialmente una reducción de personal).
- ❖ La reducción de variabilidad disminuye la necesidad de recursos.
- ❖ La reducción de variabilidad permite disminuir las tareas de kanban, el WIP y resulta en un tiempo de entrega promedio menor.
- ❖ La holgura permite oportunidades de mejora.
- ❖ Las mejoras de proceso llevan a mayor productividad y mayor previsibilidad.

❖ CAPÍTULO 4 ❖

Del peor al mejor en cinco cuartos

En octubre de 2004, Dragos Dimitriu era un director de programa en Microsoft. Había tomado cargo recientemente de un departamento que tenía la reputación de ser el peor en la división de TI de Microsoft.

El título “director de programa” en Microsoft es interpretado en otros lugares como director de proyecto, pero ahí también incluye algunas responsabilidades de análisis y arquitectura. A un director de programa le es asignado una iniciativa, proyecto, o producto y tiene responsabilidad sobre una característica o un conjunto de características. El director de programa recluta entonces recursos de áreas funcionales tales como desarrollo y pruebas con el fin de completar el trabajo. En el caso de Dragos, él era responsable del mantenimiento de software de la unidad de negocio XIT. Este equipo (mostrado en la Figura 4.1), con base en un proveedor en India con clasificación de nivel 5 del modelo CMMI, contaba con 3 desarrolladores, 3 ingenieros de pruebas, y directores locales. Desarrollaba actualizaciones menores y corregía defectos en producción para cerca de 80 aplicaciones entrecruzadas funcionales de IT utilizadas por personal de Microsoft en todo el mundo. Dragos se encontraba en el campus corporativo en Redmond, Washington. En ese entonces, yo también trabajaba allí.



Figura 4.1 El equipo en Hyderabad, India, a finales de 2004; Dragos es la cuarta persona a la izquierda

El Problema

Dragos se ofreció como voluntario para hacerse cargo del equipo que tenía la peor reputación en servicio a clientes dentro de Microsoft. Su trabajo como agente de cambio, determinado a resolver los tiempos de entrega largos y pobres expectativas, estaba obstaculizado por el clima político. Varios de sus predecesores en esa posición aun eran colegas trabajando en otros proyectos dentro de la misma unidad de negocio y les preocupaba que si él mejoraba el rendimiento, los haría quedar mal en comparación.

Los programadores y ingenieros de pruebas que trabajaban para el vendedor estaban siguiendo la metodología Proceso de Software Personal/Proceso de Software de Equipo (en inglés *Personal Software Process/Team Software Process—PSP/TSP*) del Software Engineering Institute. Microsoft demandaba esto de manera contractual. Jon De Vaan, quien en aquel entonces le reportaba directamente a Bill Gates, era un gran fanático de Watts Humphrey del Software Engineering Institute. Como jefe de Excelencia de Ingeniería en Microsoft él estaba en la posición de poner obligaciones sobre los procesos utilizados dentro del departamento de TI y por sus vendedores. Esto significaba que cambiar el método del ciclo de vida del desarrollo de software no era una opción viable.

Dragos se dio cuenta que ni el método PSP/TSP, ni la clasificación CMMI del vendedor podían ser la causa raíz de sus problemas. De hecho, el equipo producía bastante de lo que se le pedía y con muy alta calidad. Sin embargo, tenían un tiempo de entrega de cinco meses para requerimientos de cambio y esto, junto con su *backlog* de solicitudes, estaba creciendo descontroladamente. La percepción era la de un equipo que está mal organizado

y administrado. Como resultado, la alta dirección no estaba dispuesta a dar dinero adicional para corregir el problema.

Por ende, las restricciones de cambio eran políticas, físicas y relacionadas con políticas empresariales. El me pidió consejo.

Visualice el flujo de trabajo

Le pedí a Dragos que dibujara el flujo de trabajo. El dibujó un diagrama simple que describía el ciclo de vida de una solicitud de cambio y conforme lo hizo, discutimos los problemas. La Figura 4.2 es un facsímil de lo que dibujó. El hombre de palo con GP representa a Dragos.

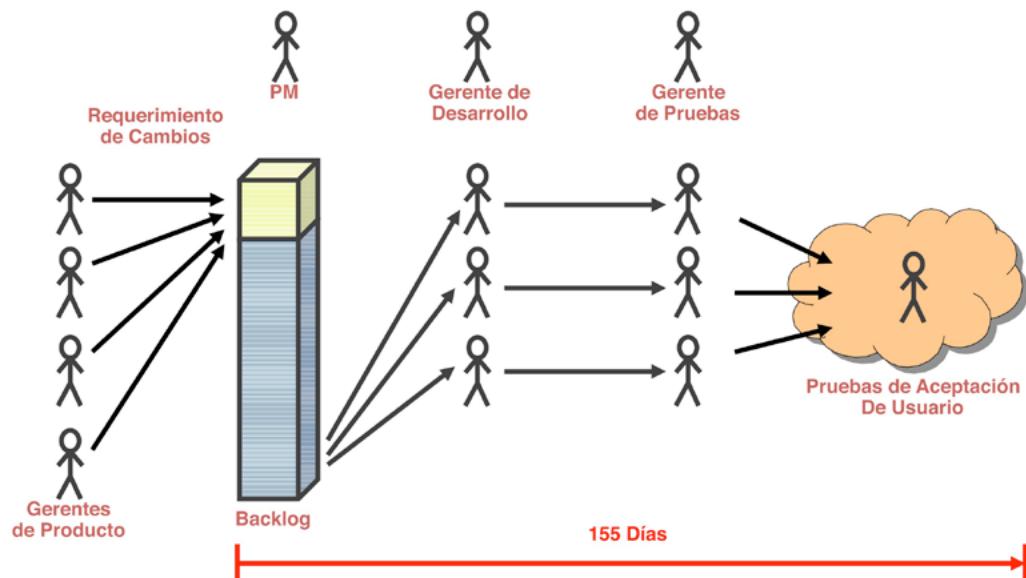


Figura 4.2 Ingeniería de MantenimientoXIT – Flujo de trabajo inicial que muestra el tiempo de entrega

Las solicitudes llegaban de manera descontrolada. Cuatro directores de producto representaban y controlaban los presupuestos para un número de clientes propietarios de aplicaciones mantenidas por XIT. Ellos agregaban nuevas solicitudes, incluyendo defectos que llegaron a producción (descubiertos en el campo). Estos defectos no habían sido generados por el equipo de mantenimiento, sino por los equipos de proyecto de desarrollo de aplicaciones. Esos equipos de desarrollo de aplicaciones eran desmantelados, en general, un mes después de la entrega de un nuevo sistema y el código fuente se le entregaba al equipo de mantenimiento.

Factores que afectan el rendimiento

Cuando cada solicitud llegaba, Dragos la enviaba a la India para ser estimada. La política era que los estimados tenían que hacerse y regresarse a los propietarios de negocio dentro de 48 horas. Esto facilitaría calcular el retorno de la inversión (ROI) y decidir si se procedería con la solicitud. Una vez por mes, Dragos se reunía con los directores de producto y otras partes interesadas para re-priorizar el *backlog* y crear un plan proyectado a partir de las solicitudes.

En este entonces, el throughput del número de solicitudes mensuales era de alrededor de siete. El *backlog* tenía 80 o más ítems en el y estaba creciendo. Esto significaba que 70 o más solicitudes estaban siendo repriorizados y reprogramados cada mes y que las solicitudes estaban llevando más de cuatro meses, en promedio, en ser procesados. Esto era la causa raíz de la insatisfacción. Estas solicitudes eran pequeñas y la continua re-priorización significaba que los solicitantes estaban continuamente decepcionados.

Las solicitudes eran seguidas mediante una herramienta llamada *Product Studio*. Una versión actualizada de esta herramienta fue entregada al público como “*Team Foundation Server Work Item Tracking*”. El equipo de mantenimiento de XIT era un tipo de organización que veo frecuentemente en mi trabajo de consultoría y enseñanza—tenían muchos datos, pero no los estaban utilizando. Dragos comenzó a minar los datos y descubrió que una solicitud promedio tomaba 11 días de ingeniería. Sin embargo, los tiempos de entrega de entre 125 y 155 días eran habituales. Más del 90 por ciento del tiempo de entrega tenía que ver con colas, u otras formas de desperdicio.

Las estimaciones para los nuevos trabajos que llegaban consumían mucho esfuerzo. Decidimos analizar esto mediante elucubración. A pesar de ser conocido como estimación de “orden de magnitud aproximado” (del inglés: Rough Order of Magnitude, ROM), la expectativa del cliente era tener una estimación muy precisa y los miembros del equipo habían aprendido a tener mucho cuidado en prepararlas. Cada estimación le estaba tomando alrededor de un día a cada desarrollador y ingeniero de prueba. Rápidamente calculamos que tan solo el esfuerzo de estimación estaba consumiendo alrededor del 33 por ciento de la capacidad y que en un mes malo podría llegar a ser hasta del 40 por ciento. Esta capacidad era asignada en preferencia sobre el trabajo de codificación y pruebas. La estimación de nuevas solicitudes también tendía a cambiar aleatoriamente los planes hechos para ese mes.

Adicionalmente a las solicitudes de cambio, el equipo tenía un segundo tipo de trabajo conocido como cambios de texto en producción (CTP), los cuales eran generalmente de naturaleza textual o gráfica, o involucraban modificaciones de valores en tablas o archivos XML. Estos cambios no requerían de un desarrollado y eran efectuados por los propietarios de negocio, directores de producto, o el director de programa, pero requerían de un paso de prueba formal por lo que afectaba a los ingenieros de pruebas.

Los CTPs llegaban sin mucha advertencia y eran, tradicionalmente, apresurados sobre cualquier otro trabajo o esfuerzo de estimación. Los CTPs tendían a llegar en lotes espóradicos y generaban aleatoriedad a cualquier plan hecho para ese mes (ver la Figura 4.3).

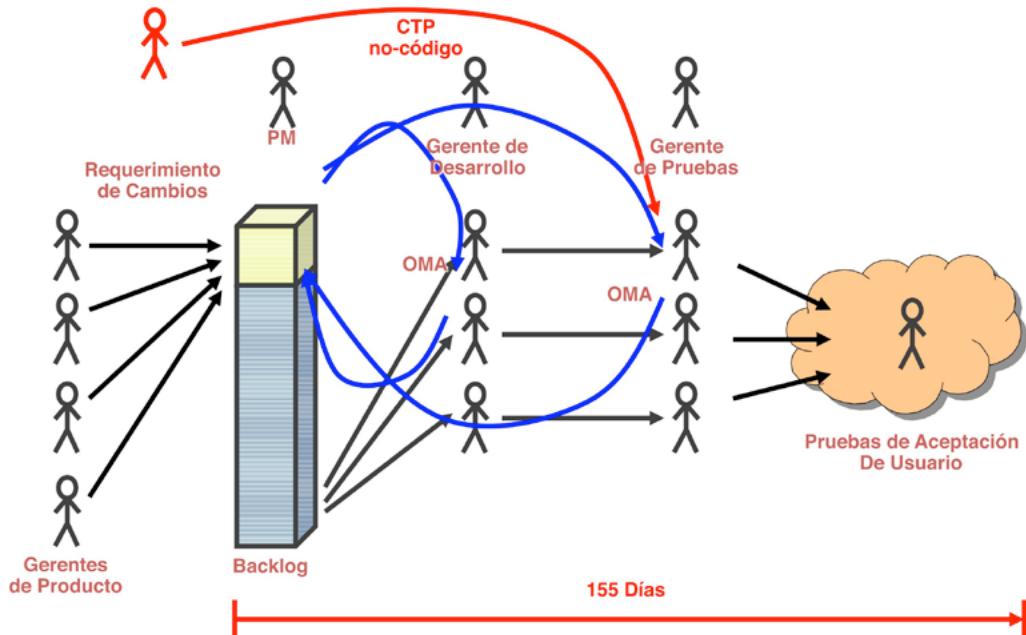


Figura 4.3 Flujo de trabajo que muestra las estimaciones OMA y la entrada de CTP

Hacer las políticas de proceso explícitas

El equipo estaba siguiendo el proceso requerido que incluía muchas decisiones basadas en malas políticas que habían sido formuladas por directores a varios niveles. Es importante considerar un proceso como un conjunto de políticas que gobiernan el comportamiento. Estas políticas están bajo el control de la dirección. Por ejemplo, la política sobre el uso de PSP/TSP había sido establecida por el nivel de vicepresidencia ejecutiva, un peldaño bajo Bill Gates y esta política sería muy difícil o imposible de cambiar. Sin embargo, muchas otras políticas, tales como la política de priorizar estimaciones sobre la codificación y pruebas mismas, eran desarrolladas localmente estaban bajo la autoridad colaborativa de los directores inmediatos. Es posible que las políticas hicieran sentido al momento de ser implementadas; pero las circunstancias cambian y no se había hecho ningún intento por revisar y actualizar las políticas que gobiernan la operación del equipo.

La estimación era un desperdicio

Después de alguna discusión con sus colegas y su director, Dragos decidió promulgar dos cambios iniciales a nivel administrativo. Primero, el equipo dejaría de estimar. El deseaba recuperar la capacidad desperdiciada por la actividad de estimación y utilizarla para desarrollar y probar software. Eliminar la aleatoriedad en el calendario generada por la estimación también mejoraría la previsibilidad y la combinación tendría, esperanzadoramente, un gran impacto en la satisfacción del cliente.

Sin embargo, eliminar la estimación era problemático. Afectaría el cálculo del ROI (retorno de inversión) y a los clientes les podría preocupar que malas elecciones de prioridad fueran hechas. Adicionalmente, las estimaciones eran usadas para facilitar contabilidad y transferencias de presupuesto entre departamentos. Las estimaciones también eran utilizadas para implementar políticas de gobierno. Solamente las solicitudes pequeñas eran permitidas a través del sistema de mantenimiento. Las solicitudes grandes, aquellas que excedían de 15 días de desarrollo o prueba, tenían que ser sometidas a una iniciativa de proyecto mayor y pasar por el proceso de gobierno de gestión de cartera de la oficina de gestión de programas (en inglés: Program Management Office – PMO). Visitaremos estos asuntos de nuevo dentro de poco.

Limitar el trabajo-en-progreso

El otro cambio que Dragos decidió hacer fue limitar el trabajo-en-progreso y tomar trabajo de una cola de entrada conforme el trabajo actual era completado. El escogió limitar el WIP en el desarrollo a una solicitud por desarrollador y usar una regla similar para los ingenieros de pruebas. El insertó una cola entre el desarrollo y las pruebas con el fin de recibir los CTPs y suavizar el flujo de trabajo entre desarrollo y pruebas, como se muestra en la Figura 4.4. Este enfoque de utilizar un buffer para suavizar la variabilidad en tamaño y esfuerzos está discutido en el capítulo 19.

Nota: Ésta es una elección de política. Una solicitud de cambio por desarrollador en un momento dado es una política. Puede modificarse mas tarde. Pensar sobre un proceso como un conjunto de políticas es un elemento clave del método Kanban.

Estableciendo una cadencia de entrada

Nota: Cadencia es un concepto en el Método de Kanban que determina el ritmo de un tipo de evento. Priorización, liberación, retrospectivas y cualquier evento recurrente puede tener su propia cadencia.

A fin de facilitar la decisión de limitar el WIP e instituir un sistema de arrastre, Dragos tuvo que pensar acerca de la cadencia para interactuar con los directores de producto. El pensó que una reunión semanal sería factible debido

a que el tema de la reunión sería simplemente a provisionar la cola de entrada a partir del *backlog*. En una semana típica habrían tres espacios libres en la cola, por lo que la discusión se centraría alrededor de la pregunta, “¿Cuales son los tres ítems que más les gustaría que fueran entregados a continuación?” Esta cadencia está modelada en la Figura 4.5.

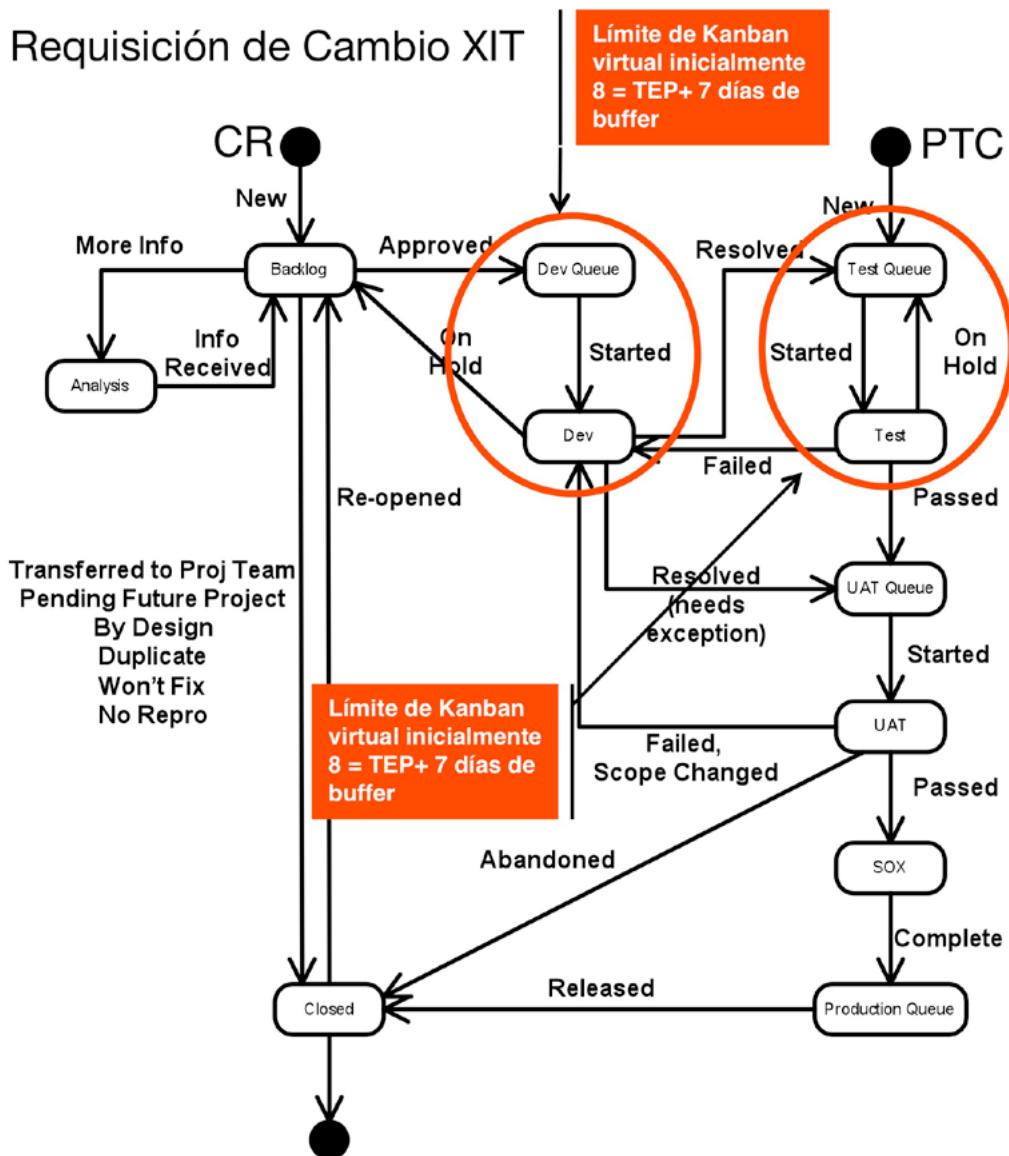


Figura 4.4 Diagrama de estado modelando el flujo de trabajo deseado con límites del WIP

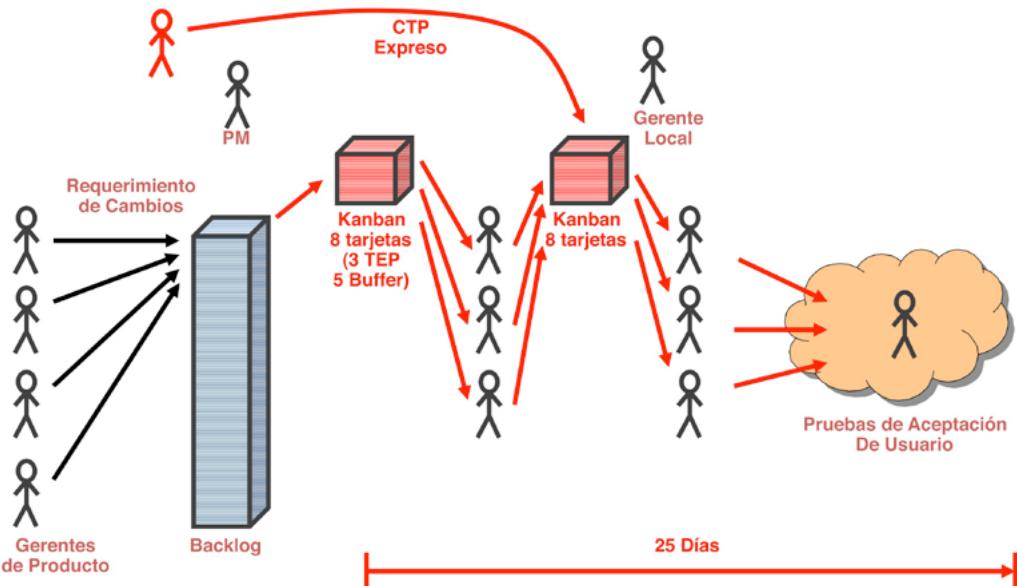


Figura 4.5 Flujo de trabajo con límites de WIP Kanban y colas

Él deseaba ofrecer un tiempo de entrega “garantizado” de 25 días desde el punto de aceptación en la cola de entrada. Este plazo de 25 días era considerablemente mayor al promedio de 11 días de ingeniería requeridos para completar el trabajo. Los esquemas estadísticos requerían alrededor de 30 días, pero él anticipaba muy pocos de ellos por lo que 25 días sonaba atractivo, especialmente comparado con el tiempo de entrega actual de alrededor de 140 días. Esperaba lograr ese objetivo con regularidad, generando confianza por parte de los directores de producto y de sus clientes conforme avanzara.

Obteniendo una nueva ganga

Dragos les llevó una oferta a los directores de producto. Les pidió que aceptaran reunirse una vez a la semana para efectuar la priorización, les informó que el iba a limitar el WIP y que su equipo dejaría de estimar. A cambio de ello, él les garantizaba entregas dentro de 25 días y les reportaría el rendimiento contra esa métrica.

Es decir, a los clientes se les estaba pidiendo que rindieran los cálculos de ROI y las transferencias de presupuesto entre departamentos basados en el esfuerzo estimado por solicitud. A cambio, se les estaba ofreciendo tiempos de entrega y confiabilidad sin precedente. Para poder ir alrededor de los asuntos de contabilidad se les estaba pidiendo que todas las solicitudes tomaran 11 días de ingeniería en promedio como se observó a

partir de datos históricos y descritos en la sección previa titulada “*Factores que afectan al rendimiento*”. Se les pedía que aceptaran que los costos fueran esencialmente fijos. Se les pedía que ignoraran el paradigma de contabilidad-de-costo en el que las transferencias de presupuesto entre departamentos estaba basada.

La explicación para justificar esto era que el vendedor tenía un contrato de 12 meses facturado mensualmente. El vendedor asignaba gente basado en el contrato y esas personas era pagadas independientemente de si trabajaban o no. El presupuesto para financiar esto desde los cuatro directores de producto era simplemente un importe fijo, distribuido proporcionalmente. Dragos garantizó que cada uno de los directores de producto obtendría una asignación equitativa de capacidad. Esto los liberaría de tener que rastrear solicitudes individuales. Si pudieran aceptar que sus dólares les compraban capacidad y que tal capacidad estaba garantizada, tal vez ignorarían su tendencia a hacer transferencias de presupuesto y gestionar el costo basado por unidad. Algunas reglas simples fueron creadas para determinar quién sería escogido para llenar la cola tal que la capacidad fuera asignada de manera justa. Para lograr esto fue suficiente utilizar un esquema ponderado de *round-robin*.

Implementando los cambios

Mientras que los directores de producto y muchos de los colegas gerenciales de Dragos en la unidad de XIT se mantenían escépticos, el consenso era que lo intentara. Después de todo, las cosas estaban mal y seguían empeorando. ¡Con certeza esto no lo haría todavía peor de lo que ya era! Alguien tenía que intentar algo y se esperaba que Dragos implementara algunos cambios.

Los cambios fueron promulgados.

Comenzó a funcionar. Las solicitudes eran procesadas y entregadas a producción. Los plazos de entrega de los nuevos compromisos se cumplieron dentro de los 25 días prometidos. Las reuniones semanales funcionaron sin problemas y la cola era renovada cada semana. La confianza en los directores de producto comenzó a generarse.

Ajustando políticas

Usted ha de preguntarse cómo la priorización era efectuada si ya no se hacían cálculos de ROI. Sucedé que no era necesario. Si algo era importante y de alto valor, era elegido del *backlog* para ponerlo en la cola; y si no lo era entonces no era elegido. Algún tiempo después Dragos reconoció que una

Nota: Este es un tema común en el Método Kanban. La combinación de políticas explícitas, transparencia y visualización facultan a los miembros individuales de los equipos a tomar sus propias decisiones y gestionar riesgos ellos mismos. La gerencia llega a tener confianza en el sistema porque entienden que el proceso está hecho de políticas. Las políticas están diseñadas para gestionar riesgo y entregar las expectativas del cliente. Las políticas son explícitas, el trabajo es rastreado transparentemente y todos los miembros del equipo entienden las políticas y como usarlas.

nueva política era necesaria: Cualquier ítem de más de seis meses de antigüedad era purgado del *backlog*. Si no era lo suficientemente importante para ser seleccionado dentro de seis meses de su arribo, podía asumirse que no era tan importante después de todo. Si en realidad si lo era entonces era reenviado.

¿Qué se tenía en la política de gobierno para prevenir que ítems grandes se colaran a través de mantenimiento cuando deberían en su lugar ser enviados a un proyecto mayor? Esto fue resuelto aceptando el hecho de que algunos se colarían. Los datos históricos indicaban que se trataba de menos del dos por ciento de las solicitudes. Se les instruyó a los desarrolladores que estuvieran alerta; si una nueva solicitud que comenzaran a desarrollar pareciera ser grande y estimaban que requeriría de más de 15 días de esfuerzo entonces debería alertar a su director local. El riesgo y costo de hacer esto era menor a la mitad del uno por ciento de la capacidad disponible. Era una gran compensación. Eliminando estimaciones, el equipo ganó más del 33 por ciento de su capacidad bajo el riesgo de gastar menos del 1 por ciento de esa capacidad. ¡Esta nueva política apodero a los desarrolladores a gestionar el riesgo y a hablar al respecto cuando fuera necesario!

Se le dio seis meses a los primeros dos cambios para que se asentaran. Como se mencionó, una política para sondear el *backlog* fue agregada; la reunión mensual con los propietarios de producto también desapareció. El proceso estaba corriendo de manera tan suave que Dragos hizo que la herramienta *Product Studio* se modificara de manera tal que le enviará un email cuando un espacio se hiciera disponible en la cola. El entonces alertaría a los propietarios de producto por medio de correo electrónico, quienes decidirían entonces quien de ellos sería el siguiente en elegir. Se haría una elección y una solicitud del *backlog* se usaría para aprovisionar la cola dentro de dos horas después de que el espacio se hubiera hecho disponible.

Buscando por más mejoras

Dragos comenzó a buscar por más mejoras. Había estado estudiando los datos históricos sobre la productividad de los ingenieros de pruebas en su equipo y lo comparó con otros equipos dentro de XIT suministrados por el mismo vendedor. El sospechaba que los ingenieros de pruebas no tenían la suficiente carga de trabajo y tenían mucha capacidad de holgura. Por implicación los desarrolladores eran un cuello de botella significativo. El decidió visitar el equipo en India. A su regreso le instruyó al vendedor que hiciera un cambio en la asignación de personal. Redujo el equipo de prueba de tres a dos y agregó un desarrollador (ver Figura 4.6). Esta acción resultó en un incremento casi lineal de la productividad, con un incremento del throughput en ese cuarto fiscal de 45 a 56.

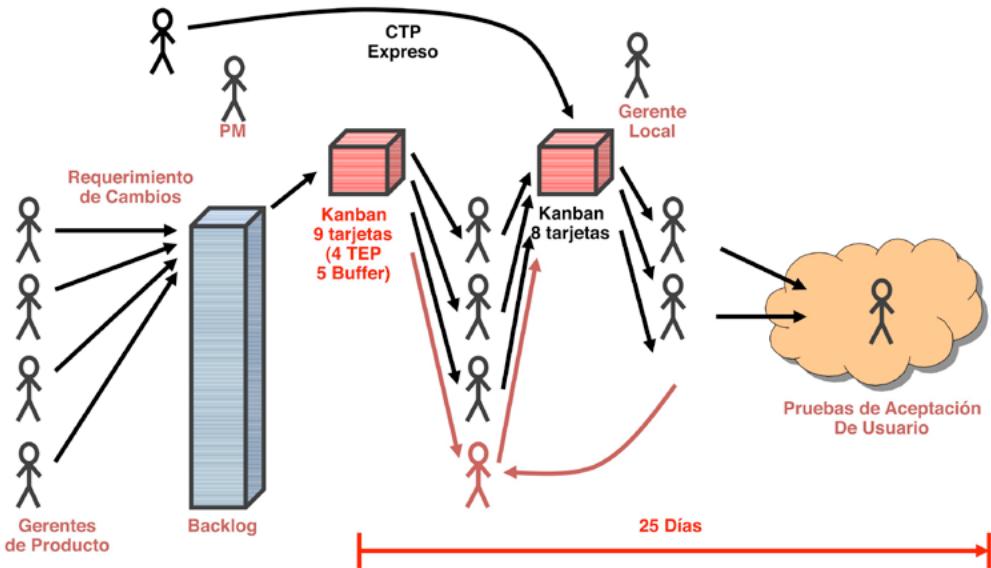


Figura 4.6 Nivelando recursos

El año fiscal de Microsoft estaba finalizando. La alta dirección notó la mejora significativa de la productividad y la consistencia de las entregas por parte del equipo de Ingeniería de mantenimiento de software del XIT. Finalmente, la dirección confió en Dragos y las técnicas que estaba empleando. Se le asignó suficiente dinero al departamento para agregar dos personas más. Un desarrollador y un ingeniero de prueba adicionales fueron agregados en Julio de 2005. Los resultados fueron significativos (ver Figura 4.7).

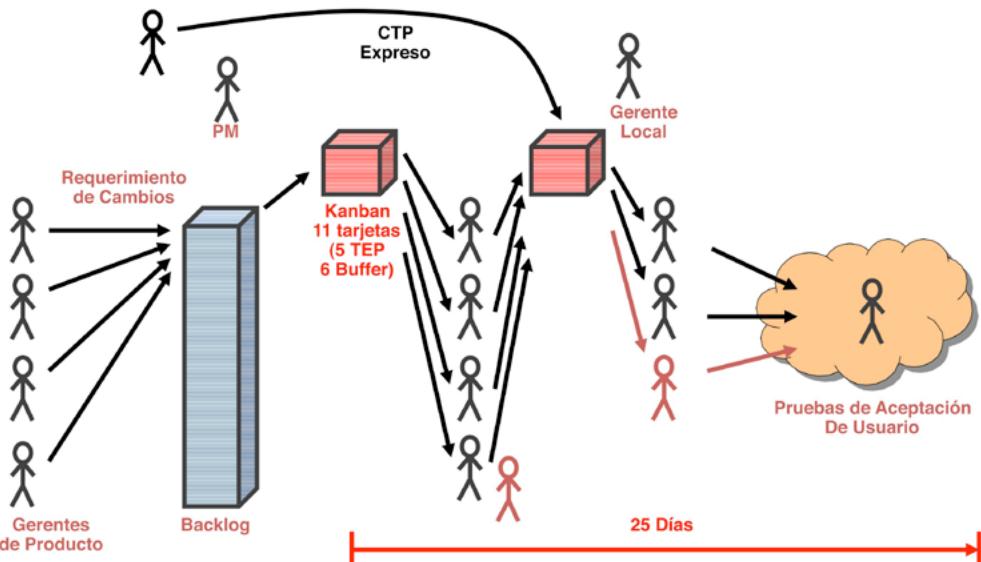


Figura 4.7 Agregando recursos adicionales

Resultados

La capacidad adicional fue suficiente para incrementar el throughput más allá de la demanda. ¿El resultado? El *backlog* fue eliminado enteramente en Noviembre 22 de 2005. Para ese entonces el equipo había reducido el tiempo de entrega a un promedio de 14 días frente a un tiempo de ingeniería de 11 días. El rendimiento de la fecha límite dentro del tiempo meta de entrega de 25 días era del 98 por ciento. El rendimiento de procesamiento de solicitudes había crecido más del triple, mientras que los tiempos de entrega se redujeron más del 90 por ciento y la fiabilidad mejoró de manera semejante. No hubo cambios en el proceso de desarrollo o de pruebas. La gente que trabajaba en Hyderabad no era conscientes de ningún cambio significativo. El método PSP/TSP no fue modificado y todos los requerimientos de gobierno corporativo, proceso y contractuales de vendedor se cumplieron completamente. El equipo ganó el Premio de Excelencia de Ingeniería en la segunda mitad de 2005. A Dragos se le apremió con más responsabilidades y la administración diaria del equipo se le dió al director local de línea en India, quién se trasladó a Washington.

Estas mejoras llegaron en parte debido a la increíble habilidad técnica de Dragos Dumitriu, pero los elementos básicos de mapear la cadena de valor, analizar el flujo, establecer límites de WIP, e implementar un sistema de arrastre fueron herramientas clave. Sin el paradigma de flujo y enfoque de kanban de limitar el WIP, las ganancias de rendimiento no habrían sido posibles. Kanban permitió cambios incrementales con poco riesgo político y poca resistencia al cambio.

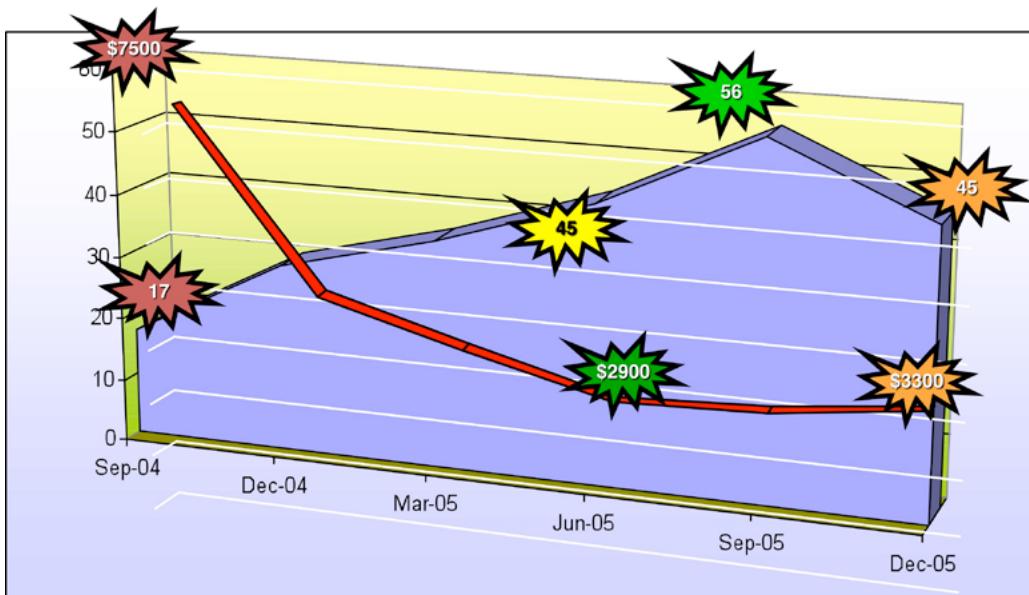


Figura 4.8 Throughput Trimestral empalmado con el Costo Unitario

El caso del XIT muestra como un sistema de arrastre con WIP limitado fue implementado en un proyecto distribuido utilizando recursos de *offshore* y facilitado con una herramienta electrónica de seguimiento. No se tenía un tablero visual y muchas de las características sofisticadas del Método Kanban descritas en este libro aún estaban por surgir. Sin embargo, ¿que director podría ignorar la posibilidad de que una productividad de más d200 por ciento, con un reducción del tiempo de entrega del 90 por ciento, pueda lograrse con previsibilidad significativamente mejorada y con mínimo riesgo político y resistencia al cambio?

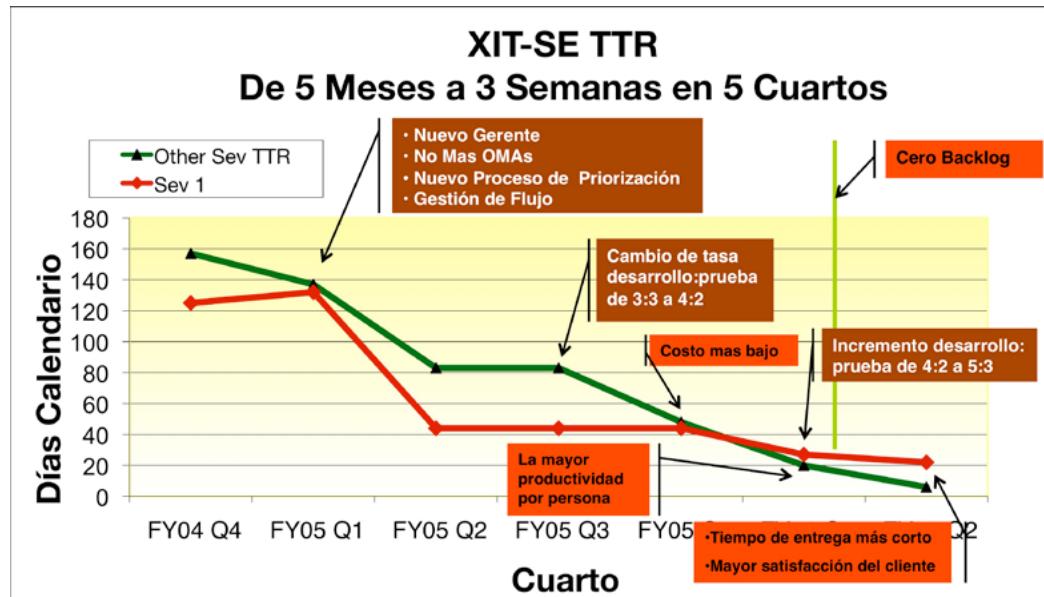


Figura 4.9 Tiempo Sustentable para Resolver, mostrado en años fiscales de Microsoft

Para llevar

- ❖ El primer sistema kanban fue implementado en el equipo de mantenimiento de software de Ingeniería Sostenida del XIT en Microsoft, comenzando en 2004.
- ❖ El primer sistema kanban utilizó una herramienta electrónica de seguimiento.
- ❖ El primer sistema kanban fue implementado con un sistema de un vendedor *offshore* con nivel 5 del modelo CMMI en Hyderabad, India.
- ❖ El flujo de trabajo debe ser dibujado y visualizado.
- ❖ El proceso puede ser descrito como un conjunto explícito de políticas.
- ❖ Kanban permite cambios incrementales.
- ❖ Kanban permite cambios con riesgo político reducido.
- ❖ Kanban permite cambios con resistencia mínima.
- ❖ Kanban mostrará oportunidades de mejora que no involucran cambios complejos en los métodos de ingeniería.
- ❖ El primer sistema kanban produjo una mejora de productividad mayor al 200 por ciento, un 90 por ciento de reducción de tiempo de espera y una mejora similar en previsibilidad.
- ❖ Mejoras significativas son posibles mediante la administración de cuellos de botella, eliminación de desperdicio y la reducción de variabilidad que afectan a las expectativas y satisfacción de los clientes.
- ❖ Los cambios pueden tomar tiempo en alcanzar su efecto total. Este primer caso de estudio requirió 15 meses.

❖ CAPÍTULO 5 ❖

Una cultura de mejora continua

La palabra Japonesa *kaizen* significa literalmente “mejora continua”. La cultura en un lugar de trabajo donde toda la fuerza de trabajo está enfocada en mejorar la calidad, la productividad y la satisfacción del cliente continuamente se le conoce como una “cultura Kaizen”. De hecho muy pocos negocios han alcanzado tal cultura. Empresas tales como Toyota, donde la participación de los empleados es su programa de mejora es cerca del 100 por ciento y donde, en promedio, una sugerencia de cada empleado es implementada cada año como parte de mejora continua, son raras.

En el mundo del desarrollo de software, el Software Engineering Institute (en inglés: Software Engineering Institute – SEI) de la Universidad Carnegie Mellon ha definido el nivel más alto de su Integración del Modelo de Madurez de Capacidad (CMMI) como Optimizando. Optimizando implica que la calidad y el rendimiento de la organización está siendo refinada continuamente. Aunque no está indicado explícitamente, debido a que el CMMI dice muy poco respecto a cultura, como una organización, lograr un comportamiento de optimización es más probable que suceda dentro de una cultura Kaizen.

La Cultura Kaizen

Para entender por qué es tan difícil alcanzar una cultura Kaizen, debemos entender primero como se vería tal cultura. Solo entonces podremos discutir por qué desearíamos lograr tal cultura y cuáles serían sus beneficios.

En la cultura Kaizen la fuerza de trabajo está facultada. Los individuos se sienten libres para tomar acción; libres de hacer lo que es correcto. Espontáneamente se enjambran sobre los problemas, discuten opciones, e implementan soluciones y mejoras. En una cultura kaizen la fuerza de trabajo no tiene temor. La norma subyacente es que la dirección sea tolerante a las fallas si la experimentación e innovación es para obtener progreso y mejoras de rendimiento. En una cultura kaizen los individuos tienen la libertad (dentro de ciertos límites) de auto-organizarse alrededor del trabajo que hacen y como lo hacen. Los controles y señales visuales son evidentes y las tareas de trabajo son tomadas de manera voluntaria en lugar de asignadas por un superior. Una cultura kaizen involucra un nivel de colaboración y una atmósfera colegial donde todos cuidan del rendimiento del equipo y el negocio sobre el rendimiento propio. Una cultura kaizen se enfoca en pensar al nivel del sistema mientras se realizan mejoras que mejoran el rendimiento general.

Una cultura kaizen tiene un alto nivel de capital social. Es una cultura de alta confianza en la que los individuos, sin importar su posición en la jerarquía de toma de decisiones del negocio, se respetan los unos a los otros y la contribución de cada persona. Culturas de alta confianza tienden a tener estructuras más planas que las culturas de baja confianza. Es el grado de facultad que permite una estructura más plana para trabajar eficientemente. Por ende, lograr una cultura kaizen puede habilitar la eliminación de capas de dirección innecesarias y como resultado reducir los costos de coordinación.

Muchos aspectos de la cultura kaizen se oponen a normas sociales y culturales establecidas en la cultura occidental moderna. En el Occidente, somos educados para ser competitivos. Nuestros sistemas escolares fomentan la competitividad en lo académico y lo atlético. Incluso nuestro equipo de deporte tiende a fomentar el desarrollo de héroes los equipos se desarrollan alrededor de uno o dos jugadores excepcionalmente talentosos. La norma social es enfocarse en lo individual primero y confiar en los individuos destacados para lograr la victoria o salvarnos del peligro. No es sorprendente que luchemos en el lugar de trabajo para fomentar un comportamiento colegial, pensar a nivel de sistemas y la cooperación.

Kanban acelera la capacidad y la madurez organizacional

El Método Kanban está diseñado para minimizar el impacto inicial de los cambios y reduce la resistencia para adoptar cambios. Adoptar Kanban debería cambiar la cultura de su organización y ayudarla a madurar. Si la adopción se hace correctamente, la organización se

transformará en una que adopta cambio con facilidad y es buena en la implementación de cambios y mejoras de proceso. El SEI se refiere a esto como una capacidad de Innovación y Despliegue Organizacional (en inglés: Organizational Innovation and Deployment – OID)¹⁵ dentro del modelo CMMI. Se ha demostrado que las organizaciones que logran este alto grado de capacidad en la gestión de cambios pueden adoptar métodos de desarrollo Ágil, tales como Scrum, más rápido y mejor que organizaciones menos maduras.

Cuando usted implemente Kanban por primera vez estará buscando optimizar los procesos existentes y cambiar la cultura organizacional más que cambiar los procesos existentes por otros que puede proporcionar mejoras económicas dramáticas. Esto a llevado al criticismo¹⁷ de que Kanban tan solo optimiza algo que necesita ser cambiado. Sin embargo, se cuenta ya con considerable evidencia empírica¹⁸ de que Kanban acelera el logro de altos niveles de madurez y capacidad organizacional en áreas clave de proceso de alta madurez tales como Resolución y Análisis de Causa (en inglés *Causal Analysis and Resolution – CAR*) e Innovación y Despliegue Organizacional.

Cuando escoja utilizar Kanban como un método para dirigir el cambio en su organización, usted se estará sumando a la opinión de que es mejor optimizar lo que ya existe, porque eso es más fácil y rápido y encontrará menor resistencia que si ejecuta una iniciativa de cambio gestionada, diseñada, y nombrada. Introducir un cambio radical es más difícil que una mejora incremental. Usted debe entender también que los aspectos de juego colaborativo de Kanban contribuirán a un cambio significativo en la cultura y madurez de su empresa. Este cambio le permitirá cambios mucho más significativos más adelante, de nuevo con menor resistencia que si intentara hacer esos cambios inmediatamente. Adoptar Kanban es una inversión en la capacidad, madurez y cultura de su organización. No se pretende ser una solución rápida.

CASO DE ESTUDIO: DESARROLLO DE APLICACIONES EN CORBIS

Cuando introduje un sistema kanban en Corbis, en 2006, lo hice por muchos de los beneficios mecánicos que fueron demostrados en el XIT de Microsoft en 2004 (como lo describí en el capítulo 4). La aplicación inicial era la misma —mantenimiento de aplicaciones de TI. Yo no estaba anticipando un cambio cultural significativo o un cambio significativo en la madurez organizacional. No esperaba que de este trabajo surgiera lo que ahora conocemos como el Método Kanban.

Conforme escribo este libro en 2010, se ha establecido que Kanban se ajusta naturalmente al trabajo de mantenimiento de TI. En 2006 no era claro, pero una forma de sistema kanban parecía encajar bien con los problemas funcionales del trabajo de mantenimiento. No me fui a Corbis con la intención de “hacer Kanban”. Fui allí con la intención de mejorar la satisfacción de clientes desde la perspectiva del equipo de desarrollo de software. Fue una coincidencia feliz que el primer problema

a atender fue la carencia de previsibilidad concerniente a las entregas del grupo de mantenimiento de software de TI.

ANTECEDENTE Y CULTURA

En 2006, Corbis era un negocio privado con cerca de 1,300 empleados en todo el mundo. El controlaba los derechos digitales de muchas obras fascinantes de arte y también representaba a aproximadamente 3,000 fotógrafos profesionales, las licencias de su trabajo para ser utilizado por editores y anunciantes. Era la segunda empresa más grande de surtido de fotografías en el mundo. Contaba también con otras líneas de negocio; de las cuales la más notable era el negocio de licencias de derechos que controlaba los derechos de las imágenes y nombres de personalidades y celebridades, a nombre de las familias, bienes raíces y firmas de gestión. El departamento de TI consistía de cerca de 110 personas entre ingeniería de software y mantenimiento de operaciones/sistemas de red. La fuerza de trabajo se aumentaba de cuando en cuando con personal contratado para trabajar en proyectos grandes. Durante su cúspide en 2007, el departamento de ingeniería de software empleaba a 107 personas, incluyendo 35 personas de contingencia en Seattle y otras 30 con un vendedor en Chennai, India. La mayoría de las pruebas eran efectuadas por el equipo en Chennai. Había un enfoque de gestión de proyectos muy tradicional. Todo era planificado en un árbol de dependencia de tareas y acumulativo por una oficina de gestión de programas. Se trataba de una empresa con una cultura conservadora en la que había sido una industria relativamente conservadora y de movimiento lento. Aplicaba enfoques tradicionales y conservativos de gestión de proyectos y para el ciclo de vida de la ingeniería de software.

El departamento de TI mantenía un conjunto diverso de aproximadamente 30 sistemas. Algunos eran los habituales sistemas de contabilidad y recursos humanos; otros eran aplicaciones exóticas y a veces esotéricas, para la industria de gestión de derechos digitales. Había un rango amplio de tecnologías, plataformas de software y lenguajes soportados. La fuerza de trabajo era increíblemente leal; mucha gente en el departamento de TI había estado en la empresa durante ocho años y algunos tenían hasta 15 años de servicio. Lo cual no estaba mal para una empresa de alrededor de 17 años de antigüedad. El proceso existente era un sistema tradicional de ciclo de vida de desarrollo de software estilo Cascada, que había sido institucionalizado a lo largo de los años con la creación de un departamento de análisis de negocio, un departamento de análisis de sistemas, un departamento de desarrollo y un departamento de pruebas de *offshore*. Dentro de estos departamentos se contaba con muchos especialistas, tales como analistas cuyo antecedente era en contabilidad y cuya especialidad era finanzas. Algunos desarrolladores eran también especialistas, por ejemplo, los programadores de J. D. Edwards, quienes mantenían el software de contabilidad de J. D. Edwards.

Ninguno de estos era ideal, pero es lo que era. Las cosas eran de la forma que eran. Cuando entré a la compañía había cierta expectativa y temor de que yo impondría un método ágil y utilizaría mi poder jerárquico para forzar a la gente a cambiar de comportamiento. Aunque eso habría funcionado, habría sido brutal y el impacto durante la transición habría sido severo. Yo tenía temor de hacer las

cosas peor; temor de que los proyectos se paralizarían mientras se daba nuevo entrenamiento y el personal se adaptaba a las nuevas maneras de trabajar. También tenía temor de perder personal clave, sabiendo que la fuerza de trabajo era frágil debido a los niveles excesivos de especialización. Opté por introducir un sistema kanban, obtener el trabajo de mantenimiento de sistemas de nuevo en pista y ver qué sucedería a partir de eso.

LA NECESIDAD DE UNA FUNCIÓN DE MANTENIMIENTO DE SOFTWARE

El mantenimiento de software (o RRT “Rapid Response Team”, como se le conocía internamente) había sido fundado por el comité ejecutivo con un presupuesto adicional del diez por ciento para el departamento de ingeniería de software. Esto contaba por cinco personas adicionales en 2006. Esas cinco personas fueron contratadas algún tiempo antes de mi llegada. Debido a la naturaleza diversa de los sistemas involucrados y el grado alto de especialización en el equipo, se decidió que un equipo dedicado de cinco personas para hacer el trabajo de mantenimiento no sería una buena solución. Por lo que cinco personas adicionales fueron añadidas a la reserva general de los recursos: un director de proyectos, un analista, un desarrollador y dos ingenieros de pruebas. Esto introdujo la complicación adicional de que era necesario, desde el punto de vista de gobierno, mostrar que las cinco personas adicionales estaban de hecho haciendo trabajo de mantenimiento y no podían ser absorbidos por la cartera de proyectos grandes. Sin embargo, en cualquier día, esas cinco personas podían ser cualesquiera de las aproximadamente 55 personas en el equipo.

Una expectativa se había establecido de que un equipo de mantenimiento le permitiría a Corbis hacer entregas incrementales a los sistemas de TI cada dos semanas. Los proyectos grandes habían incluido habitualmente grandes actualizaciones del sistema y nuevas entregas de los sistemas una vez cada tres meses. Pero a medida que el negocio maduró y la naturaleza de estos sistemas se hicieron más complejos, esta cadencia de grandes entregas trimestrales se había convertido en intermitente. Además, algunos de los sistemas existentes estaban efectivamente al final de su vida y necesitaban sustitución completa. El reemplazo del sistema legado es un reto importante y habitualmente involucra proyectos grandes con mucho personal hasta que una paridad de funcionalidad es alcanzada y el sistema viejo puede ser apagado conforme el nuevo se pone en línea. (Este enfoque está en realidad muy lejos de ser óptimo, pero es habitual).

Por tanto las entregas de mantenimiento eran el área de TI dentro de Corbis donde Kanban podría permitir alguna forma de agilidad de negocio.

PEQUEÑOS PROYECTOS PARA MANTENIMIENTO NO ESTABAN FUNCIONANDO

El sistema existente para entregar versiones de mantenimiento, el sistema que estaba descompuesto, era para planificar una serie de proyectos cortos de dos semanas de longitud. Esto habría sido reconocible como desarrollo ágil de software utilizando iteraciones de dos semanas, pero no lo era. Cuando recién llegué, la negociación del alcance para un ciclo de entrega de dos semanas

estaba tomando alrededor de tres semanas. Por lo que los costos de transacción de la parte frontal de una entrega eran mayores que el trabajo de valor agregado. Estaba tomando alrededor de seis semanas el poner una entrega de dos semanas fuera de la puerta.

IMPLEMENTANDO CAMBIO

Aun antes de hacer cualquier cambio era claro que el status quo era inaceptable. El sistema actual no era capaz de entregar el nivel requerido de agilidad de negocio. El mantenimiento del sistema nos dio una oportunidad ideal para introducir el cambio. El trabajo de mantenimiento en general no era de carácter crítico. Pero sin embargo era altamente visible, ya que el negocio tenía entrada directa a la priorización y las elecciones eran altamente tácticas y e importantes para las metas de negocio de corto plazo. El mantenimiento del sistema es algo que a todos les importaba y deseaban que trabajara efectivamente. Y, finalmente, había una razón de peso para hacer cambios. Todo el mundo estaba descontento con el sistema existente. Los desarrolladores, ingenieros de pruebas y analistas estaban agravados todo con la pérdida de tiempo negociando el alcance y el personal de negocio estaba enormemente insatisfecho con los resultados.

Diseñamos un sistema kanban con entregas regulares cada dos semanas, planificadas para la 1:00 PM todos los segundos miércoles y con reuniones de priorización programadas con la gente de negocios, programada para las 10:00 AM todos los lunes. De esa forma la cadencia de priorización era semanal y la cadencia de entrega era de dos veces por semana. La elección de la cadencia se determinó a través de discusiones en colaboración con los socios en los niveles anteriores y posteriores del proceso y fue basado en los costos de transacción y coordinación de las actividades. Algunos otros cambios fueron hechos. Introducimos una cola de entrada llamada Listo Para Ingeniería con un límite de WIP de 5 ítems y añadimos límites de WIP a través del ciclo de vida del análisis, desarrollo, construcción y prueba del sistema. Pruebas de aceptación, escenario y listo para la producción se quedaron sin límites, ya que no se consideraron de capacidad limitada y estaban, en cierta medida, fuera de nuestro control político inmediato.

EFFECTOS PRIMARIOS DE LOS CAMBIOS

Los efectos de introducir un sistema kanban fueron, a un nivel, no sorprendentes, pero a otro nivel fueron remarcables. Comenzamos a hacer entregas cada dos semanas. Después de tres iteraciones estaban sucediendo sin incidente. La calidad era buena y había entre poca o ninguna necesidad de reparaciones de emergencia cuando el nuevo código se ponía en producción. La sobrecarga para planificar establecer horario había bajado dramáticamente y las disputas entre los equipos de desarrollo y la oficina de gestión de programa había desaparecido casi completamente. Por lo que kanban había cumplido con su promesa básica. Estábamos haciendo entregas regulares de alta calidad, con una carga gerencial mínima. Los costos de transacción y coordinación por entrega se habían reducido drásticamente. El equipo estaba obteniendo más trabajo y estaban entregando al cliente más seguido.

Fueron los efectos secundarios los que fueron aun más remarcables.

EFFECTOS NO ANTECIPADOS DE LA INTRODUCCIÓN DE KANBAN

Para el equipo de desarrollo, introdujimos la pared de tarjetas físicas utilizando notas adhesivas en un pizarrón blanco en enero de 2007. Comenzamos a llevar a cabo reuniones de pie diarias de 15 minutos frente a el tablero a las 9:30 AM. El tablero físico tuvo un gran efecto psicológico comparado con todo lo disponible de la herramienta electrónica de seguimiento que utilizamos en Microsoft. Atendiendo la reunión de pie diaria, los miembros del equipo eran expuestos a un tipo de fotografía de un lapso de tiempo del flujo de trabajo a lo largo del tablero. Los items que estaban bloqueando el trabajo eran marcados con notas color rosa y el equipo se volvió mucho más enfocado en la resolución de los asuntos y en mantener flujo. La productividad creció dramáticamente.

Con el flujo de trabajo visible en el tablero, comencé a prestarle atención a la función del proceso. Como resultado, hice algunos cambios al tablero. Mi grupo de directores llegó a entender los cambios que yo estaba haciendo y porqué los estaba haciendo y para marzo ellos mismos estaban haciendo cambios. A su vez, los miembros de sus equipos, los desarrolladores individuales, ingenieros de pruebas y analistas comenzaron a ver y entender cómo funcionaban las cosas. A principios del verano, todos en el equipo se sentían facultados para sugerir cambios y observamos la afiliación espontánea (y a menudo entre funciones) de los grupos de individuos quienes discutirían problemas y retos en el proceso y harían cambios como lo veían apropiado. Normalmente le informaban a la cadena de dirección después del hecho. Lo que emergió durante un lapso de aproximadamente seis meses fue una cultura kaizen en el equipo de ingeniería de software. Los miembros del equipo se sentían facultados. El temor fue removido. Se sentían orgullosos de su profesionalismo y de sus logros y querían hacer aún mejor.

Cambio Sociológico

Desde la experiencia en Corbis, ha habido otros reportes similares en el campo. Rob Hathaway de Indigo Blue fue el primero en realmente replicar estos resultados con el grupo de TI en IPC Media en Londres. El hecho de que otros han sido capaces de replicar los efectos sociológicos de kanban que observamos en Corbis me hace creer que hay una causa y que no fue ni coincidencia ni un efecto directo de mi involucro personal.

He pensado mucho sobre lo que provocó estos cambios sociológicos. Los métodos ágiles nos han ofrecido transparencia en el trabajo-en-progreso por una década y aún así los equipos que siguen el Método Kanban parecen lograr una cultura Kaizen más rápida y efectivamente que los equipos habituales de desarrollo de software ágil. A menudo, los

equipos que agregan Kanban a sus métodos ágiles existentes tienen una mejora significativa en capital social entre los miembros del equipo. ¿Porqué sucede esto?

Mi conclusión es que Kanban provee transparencia en el trabajo y también en el proceso (o el flujo de trabajo). Provee visibilidad en cuanto a cómo el trabajo es pasado de un grupo a otro. Kanban le permite a toda persona interesada ver los efectos de sus acciones o falta de acción. Si un ítem está bloqueado y alguien es capaz de desbloquearlo, Kanban lo refleja. Tal vez hay un requerimiento ambiguo. Normalmente , el experto en la materia que puede resolver la ambigüedad esperaría recibir un correo electrónico con una petición de reunión. Después de una llamada telefónica de seguimiento, arreglan una reunión que se ajusta a sus calendarios, tal vez tres semanas después. Con Kanban y la visibilidad que provee, el experto en la materia se da cuenta del efecto de la falta de acción y prioriza la reunión, tal vez modificando su calendario para poder tener la reunión en esa misma semana en lugar de retrasarla por dos semanas más.

En adición a la visibilidad del flujo del proceso, los límites del trabajo-en-proceso también obligan que interacciones desafiantes se lleven a cabo más pronto y más seguido. No es fácil ignorar un ítem bloqueado y simplemente trabajar en algo más. Este aspecto de “detener la línea” de Kanban parece motivar comportamiento de enjambre a través de la cadena de valor. Cuando personas de diferentes áreas funcionales y con diferentes títulos se congregan sobre un problema y colaboran para encontrar una solución, manteniendo entonces el flujo de trabajo y mejorando el rendimiento a nivel de sistema, el nivel de capital social y confianza en el equipo se incrementa. Con mayores niveles de confianza engendrada a través de una colaboración mejorada, el temor es eliminado de la organización.

Los límites de trabajo-en-progreso acoplados con las clases de servicio (explicados en el capítulo 11) también facultan a los individuos para tomar decisiones de agenda por sí mismos, sin supervisión o dirección gerencial. El facultad mejora el nivel de capital social demostrando que los superiores confían en que los subordinados toman decisiones de alta calidad por si mismos. Los directores se libran de tener que supervisar a los contribuidores individuales y pueden enfocar su energía mental en otras cosas tales como el rendimiento del proceso, gestión de riesgos, desarrollo de personal y mejorar la satisfacción de los clientes y los empleados.

Kanban mejora el nivel de capital social dentro del equipo grandemente. Los niveles mejorados de confianza y la eliminación de miedo alientan innovación colaborativa y solución de problemas. El efecto neto es la emergencia rápida de una cultura kaizen.

Propagación Viral de la Colaboración

Kanban claramente mejoró la atmósfera en el departamento de desarrollo de software en Corbis, pero fueron los resultados más allá de ese grupo los que fueron más remarcables.

Vale la pena reportar y analizar la manera en que la propagación viral de Kanban mejoró la colaboración en la compañía.

CASO DE ESTUDIO: DESARROLLO DE APLICACIONES EN CORBIS, CONTINUACIÓN

A las 10:00 AM de cada lunes, Diana Kolomyets, la director de proyecto responsable de la coordinación de las entregas de mantenimiento de sistemas de TI, convocaba la reunión del consejo de priorización del RRT. Los participantes de negocio eran habitualmente vicepresidentes. Ellos estaban a cargo de una unidad de negocio y le reportaban a un vicepresidente senior o director de nivel C de la compañía. Poniéndolo de otra manera, un vicepresidente le reportaba a un miembro del comité ejecutivo. Corbis aún era lo suficientemente pequeño para que tuviese sentido tener a un ejecutivo de tal rango alto atendiendo la reunión semanal. Igualmente, las elecciones tácticas hechas eran lo suficientemente importantes como para necesitar la dirección de un vicepresidente para influenciar una buena elección.

Usualmente, cada participante recibía un correo electrónico el viernes previo a la reunión. El correo indicaría algo como, "Anticipamos que habrán dos espacios vacíos en la cola la próxima semana. Por favor examinen los ítems en su *backlog* y seleccionen candidatos para discutirlos en la reunión del lunes".

NEGOCIACIÓN

Durante las primeras semanas del nuevo proceso, algunos de los participantes llegaban con la expectativa de negociar. Dirían, "Sé que hay solo un espacio libre, pero tengo dos cosas pequeñas, ¿pueden hacer ambas?" Tal negociación era raramente tolerada. Los otros miembros del comité se aseguraban que todos se ajustaran a las reglas. Responderían, "¿Cómo sabes que son pequeñas? ¿Te lo tomo a palabra?" o contra argumentar con, "Yo también tengo dos pequeñas. ¿Por qué no puedo tener mis favoritas seleccionadas?" Me refiero a este como el "Período de negociación", pues indica el estilo de negociación que se llevaba a cabo en las reuniones de priorización.

DEMOCRACIA

Después de unas seis semanas y por coincidencia alrededor del tiempo en que el equipo de desarrollo introdujo el uso del tablero físico, el comité de priorización introdujo un sistema de voto democrático. Ellos hicieron esto voluntaria y espontáneamente, pues se habían cansado de disputar. La negociación en la reunión era una pérdida de tiempo. Tomó varias iteraciones para refinar el sistema de voto y se asentó en un sistema donde cada participante tenía un voto para cada espacio libre en la cola de esa semana. Al principio de la reunión cada miembro proponía un número pequeño de candidatos para ser seleccionados. Conforme pasó el tiempo la propuesta de solicitudes se hizo más sofisticada; algunas personas llegaban con presentaciones en PowerPoint, otros con hojas

de trabajo que mostraban un caso de negocio. Más adelante escuchamos que algunos miembros estaban influenciando a sus colegas llevándolos a comer. Se estaban haciendo tratos, “Si yo voto por tu elección esta semana, vas a votar por la mía la próxima semana?” Debajo del nuevo sistema democrático de priorización el nivel de colaboración entre las unidades de negocio a nivel de vicepresidencia estaba creciendo. Aunque no nos dimos cuenta en ese entonces, el nivel de capital social a través de la empresa entera estaba creciendo. Cuando los líderes de las unidades de negocio comienzan a colaborar, parece ser que también lo hacen las personas en sus organizaciones. Siguen la guía su líder. El comportamiento colaborativo acoplado con visibilidad y transparencia conlleva un comportamiento más colaborativo. Me refiero a este período como “el Período de la Democracia”.

ABAJO LA DEMOCRACIA

La democracia estaba bien, pero después de cuatro meses parecía que la democracia había fallado en elegir al mejor candidato. Un esfuerzo considerable era gastado implementando una característica de comercio electrónico para el mercado del Este Europeo. El caso de negocio había sido estelar pero su candidatura era sospechosa desde el principio y algunos cuestionaron la calidad de los datos en el caso de negocio. Después de varios intentos esta característica había sido elegida y había sido implementada debidamente. Era una de las características más grandes procesadas a través del sistema RRT y mucha gente se involucró y lo notó. Dos meses después de su entrega, nuestro Director de Inteligencia de Negocio minó datos referentes a las ganancias generadas. Era una fracción de lo que se había prometido en el caso de negocio original y el período de recuperación estimado contra el esfuerzo realizado se calculó sería de 19 años. Debido a la transparencia que Kanban nos ofrecía, muchos de los interesados se dieron cuenta de ello y hubo una discusión sobre como capacidad valiosa había sido desperdiciada con esta elección cuando una elección mejor se podía haber hecho. Ese fue el final del Periodo de Democracia.

COLABORACIÓN

Lo que la reemplazo fue muy remarcable. Mantenga en mente que el comité de priorización consistía en su mayoría de empleados a nivel de vicepresidente y directores de la empresa. Ellos tenían una visibilidad amplia sobre aspectos del negocio que muchos de nosotros no teníamos conciencia. Por lo que al inicio de la reunión ellos preguntaban, “Diana, ¿cuál es el tiempo de entrega actual?” y ella contestaba, “Actualmente nos toma 44 días en promedio para llegar a producción”. Entonces preguntaban una pregunta sencilla: “¿Cuál es la iniciativa táctica de negocio más importante en esta compañía en 44 días a partir de ahora?” Debe haber habido alguna discusión, pero por lo general se llegaba a un acuerdo. “Oh, es nuestra campaña de marketing Europea que lanzaremos en el festival de Cannes”. “¡Grandioso! ¿Qué ítems del backlog se requieren para soportar el evento en Cannes?” Una búsqueda rápida produciría una lista de seis ítems. “Pues, hay tres espacios abiertos esta semana. Tomemos tres de los seis y los demás los hacemos la próxima semana”. Había poco debate. No había regateo o negociación y la reunión se terminaba en alrededor de 20 minutos. Me

refiero a este como el “Período de Colaboración”. Representa el nivel más alto de capital social y confianza entre las unidades de negocio que fue logrado durante mi tiempo como Director senior de Ingeniería de Software en Corbis.

El cambio cultural es quizás el mayor beneficio de Kanban

Fue interesante ver el cambio cultural emerger y ver como afectó a la compañía conforme los empleados siguieron la guía de sus vicepresidentes y comenzaron a colaborar más con sus colegas de otras unidades de negocio. Este cambio fue lo suficientemente profundo que el jefe ejecutivo recientemente asignado, Gary Shenk, me llamara a su oficina para preguntarme si tenía alguna explicación. Me dijo que había observado un nuevo nivel de colaboración y espíritu colegial en los rangos superiores de la compañía que unidades de negocio previamente antagonistas parecían estar llevándose mucho mejor. Sugirió que el proceso RRT tenía algo que ver con eso y me preguntó si tenía alguna explicación de eso. Mientras que estoy seguro de que no fui tan articulado como ahora, escribiendo este capítulo dos años después, lo convencí de que nuestro sistema kanban había mejorado grandemente la colaboración y que con eso el nivel de capital social entre todos los involucrados.

Los efectos culturales secundarios de lo que ahora reconocemos como Kanban (con “K” mayúscula) fueron inesperados y en muchas maneras contra-intuitivos. El me preguntó, “¿Porqué no estamos haciendo todos nuestros proyectos principales de esta manera?” ¿De hecho por qué? Por lo que nos pusimos a implementar Kanban en la cartera de proyectos principales. Hicimos esto porque Kanban permitió una cultura Kaizen y ese cambio cultural era tan deseable que el costo de cambiar las múltiples mecánicas de priorización, calendario, reportes y entrega que resultarían de implementar Kanban fueron considerados un precio que valía la pena pagar.

Para llevar

- ❖ *Kaizen* significa “mejora continua”.
- ❖ Una cultura kaizen es una en la que los individuos se sienten facultados , actúan sin temor, se afilian espontáneamente, colaboran, e innovan.
- ❖ Una cultura kaizen tiene un alto grado de capital social y confianza entre los individuos, sin importar su nivel en la jerarquía corporativa.
- ❖ *Kanban* provee transparencia tanto en el trabajo como en el proceso por el que el trabajo fluye.
- ❖ La transparencia de proceso permite a todos los interesados ver los efectos de sus acciones e inacciones.
- ❖ Los individuos son más dados a dedicar tiempo y colaborar cuando pueden ver el efecto que ello tendrá.
- ❖ Los límites de WIP en Kanban motivan un comportamiento de “detener la línea”.
- ❖ Los límites de WIP en Kanban alientan enjambramiento para resolver problemas.
- ❖ El incremento de colaboración que resulta de enjambrarse sobre problemas y la interacción con personas interesadas externas incrementa el nivel de capital social dentro del equipo y la confianza entre los miembros del equipo.
- ❖ Los límites de WIP en Kanban y las clases de servicio facultan a los individuos a tomar trabajo, priorizar y poner en agenda decisiones sin supervisión o dirección de un superior.
- ❖ Niveles incrementados de facultación incrementan el capital social y la confianza entre trabajadores y directores.
- ❖ El comportamiento colaborativo puede difundirse de manera viral.
- ❖ Los individuos tomarán el liderazgo de los directores.
- ❖ El comportamiento colegial colaborativo entre los líderes senior afectará el comportamiento de la fuerza de trabajo entera.

❖ PARTE TRES ❖

IMPLEMENTANDO KANBAN

❖ CAPÍTULO 6 ❖

Mapeo de la cadena de valor

Kanban es un enfoque que impulsa el cambio mediante la optimización del proceso existente. La esencia de comenzar con Kanban es cambiar lo menos posible. Usted debe resistir la tentación de cambiar de flujo de trabajo, los títulos, roles y responsabilidades y trabajo específico. Todo aquello a partir de lo cual los miembros del equipo y otros socios, participantes y partes interesadas obtienen su autoestima, orgullo profesional y ego debe permanecer sin cambios. El principal objetivo del cambio será la cantidad de trabajo-en-progreso y la interfaz así como la interacción con los niveles anteriores y posteriores del proceso en su negocio. Es decir, usted debe trabajar con su equipo tal y como existe para hacer el mapeo de la cadena de valor. Trate de no cambiarlo o inventarlo de manera idealista.

En algunas situaciones políticas, puede haber un proceso oficial que no se está siguiendo. Cuando intente hacer el mapeo de la cadena de valor, su equipo insistirá para que usted vuelva a documentar el proceso oficial, no el proceso actual se está utilizando. Usted debe resistir eso e insistir que el equipo haga el mapeo del proceso que realmente utiliza. Sin esto, será imposible utilizar una pared de tarjetas como una herramienta de visualización del proceso porque los miembros del equipo pueden utilizar la pared de tarjetas solamente si refleja lo que realmente hacen.

Definiendo un punto de inicio y de fin para el control

Es necesario decidir donde comenzar y terminar la visualización y al hacer eso definir los puntos de interface con los socios de los niveles anteriores y posteriores del proceso. Es importante gestionar esta etapa temprana de la implementación de Kanban con sensibilidad, pues pobres decisiones al inicio pueden llamar el fracaso. Los equipos exitosos han tendido a apegarse a adoptar la visualización del flujo de trabajo mediante tarjetas y limitando el WIP dentro de su propia esfera política de control y a negociar una nueva manera de interactuar con los socios de los niveles anteriores y posteriores del proceso. Por ejemplo, si usted controla la función de ingeniería o de desarrollo de software y tiene control o influencia sobre el análisis, diseño, prueba y codificación entonces genere un mapa de la cadena de valor y haga negociaciones sobre nuevos estilos de interacción con los socios de niveles anteriores quienes proveen los requerimientos, prioridades y administración de la cartera y aquellos de niveles posteriores encargados de operación de sistemas, o funciones de mantenimiento de producción. Dibujando los límites de esa manera, le estará preguntando a su equipo que adopte solamente el comportamiento de limitar el WIP. Usted no les está pidiendo a los equipos de niveles anteriores o posteriores que cambien la manera en que hacen su trabajo. No les está pidiendo que límiten su WIP e implementen un sistema de arrastre. Sin embargo, les está pidiendo que interactúen de manera diferente con ustedes—que interactúen de manera compatible con el sistema de arrastre que desea implementar.

Tipos de ítems de trabajo

Una vez que haya seleccionado el punto de inicio en el flujo de trabajo o la cadena de valor, identifique los tipos de trabajo que llegan a ese punto y cualquier otro que ya exista dentro del flujo de trabajo que necesitarán ser limitados. Por ejemplo, defectos son probablemente un tipo de trabajo que existe dentro del flujo de trabajo. Usted puede también identificar otros tipos de trabajo centrado al desarrollo tal como refactorización, mantenimiento de sistemas y actualización de infraestructura y trabajos relacionados. Para el trabajo que llega, usted puede tener tipos tales como Historia de Usuario o Caso de Uso, o Requerimiento Funcional, o Característica. En algunos casos, los tipos que lleguen pueden jerárquicos, tales como Épicas—una colección de historias de usuario.

Ítems de trabajo habituales vistos en equipos adoptando Kanban han incluido, pero no se han limitado, a los siguientes:

- Requerimiento
- Característica
- Historia de Usuario
- Caso de Uso

- Solicitud de Cambio
- Defecto en Producción
- Mantenimiento
- Refactorización
- Defecto
- Sugerencia de Mejora
- Asunto de Bloqueo

Es útil nombrar los tipos de ítem de trabajo por su fuente, tal como Requerimiento Regulatorio, Solicitud del Campo de Ventas, Solicitud de Planificación Estratégica, etcétera. Utilizando una convención de nomenclatura que muestra la fuente de la solicitud de trabajo transparente provee contexto adicional que le permite al sistema evolucionar y servir múltiples clientes.

Los tipos de ítems de trabajo tenderán a ser definidos por la fuente del trabajo, el flujo del trabajo, o el tamaño del trabajo. Por ejemplo, los CTP del ejemplo sobre Microsoft en el capítulo 4 tienen un flujo de trabajo diferente aunque la fuente es la misma que la de Solicitud de Cambios. No tiene sentido tener sistemas kanban separados para los dos tipos. El mismo equipo hace el trabajo. Es lo suficientemente fácil visualizar los tipos utilizando tarjetas de diferentes colores o diferentes renglones (carriles de nado; del inglés swim lanes) en un tablero de tarjetas. El orden de magnitud en tamaño podría ser normalmente : pequeño (algunos días); mediano (una o dos semanas); grande (un mes o más). Cada orden de magnitud debe tener su propio tipo.

Trazando una pared de tarjetas

Es habitual dibujar paredes de tarjetas para mostrar las actividades sobre el trabajo en lugar de funciones específicas o descripciones de puestos. A menudo funciones y actividades se traslanan fuertemente; por ejemplo, los analistas efectúan el análisis. Sin embargo, la convención de Kanban en proyectos de software durante los últimos varios años ha sido modelar el trabajo y no a los trabajadores, las funciones, o las entregas entre funciones.

Antes de dibujar una pared de tarjetas para visualizar el flujo de trabajo, puede tener sentido trazarlo o modelarlo. La figura 4.4 muestra un modelo muy formal, utilizando la notación de diagrama de estado, del flujo de trabajo deseado, con colas adicionadas para solicitudes de cambio y cambios de texto en producción procesados por el equipo de ingeniería sustentable de mantenimiento de XIT de Microsoft. Usted encontrará que un enfoque menos formal es perfectamente adecuado. Un dibujo con hombres de palito, similar a los mostrados en el capítulo 4, o un diagrama de flujo, o su equivalente son suficientes.

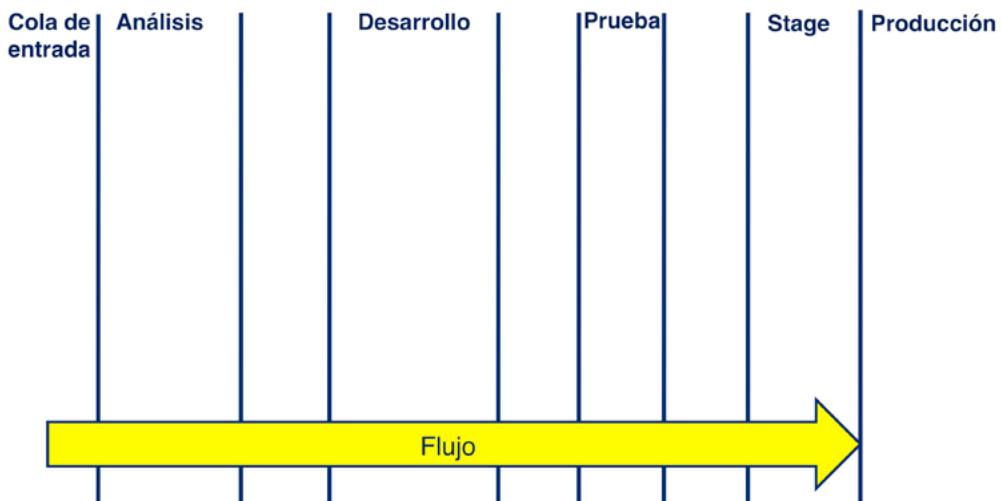


Figura 6.1 Trazo del flujo de trabajo en una pared de tarjetas (flujo de izquierda a derecha) en orden de ejecución

Una vez haya entendido su flujo de trabajo trazándolo o modelándolo, comience a definir la pared de tarjetas dibujando columnas en el tablero que representen las actividades efectuadas, en el orden en que son efectuadas, como se muestra en la Figura 6.1. Al dibujar las columnas inicialmente, tiene sentido dibujarlas con un marcador. Sin embargo, con el uso las líneas se borrará. Durante las primeras semanas usted puede que quiera hacer algunos cambios al flujo de trabajo, por lo que utilizar un marcador tiene sentido. Sin embargo, llegará un punto en el que deseará algo más permanente. Las tiendas de artículos de oficina ofrecen cintas de vinilo delgadas, diseñadas específicamente para trabajo de precisión en pizarrones blancos como se ilustra en la Figura 6.2. En Corbis se hizo común delinejar columnas y renglones en una pared de tarjetas utilizando tales cintas. Esta práctica ha sido adoptada ampliamente, con equipos utilizando diversos grados y anchos de cinta para marcar reglas y columnas.

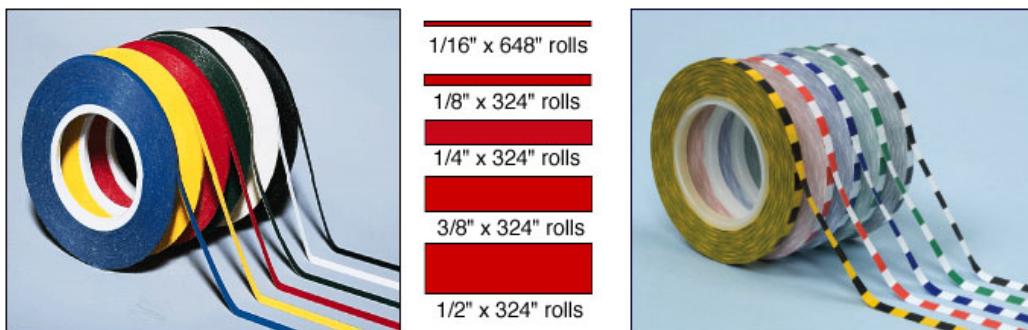


Figura 6.2 Cinta de precisión para pizarrón blanco

Note que para los pasos de actividad es necesario modelar tanto el trabajo que está en-progreso como el que ha sido completado; por convención esto se hace dividiendo la columna.

A continuación, adicione la cola de entrada y los pasos de entrega a los niveles posteriores que desee visualizar, como se muestra en la Figura 6.3.

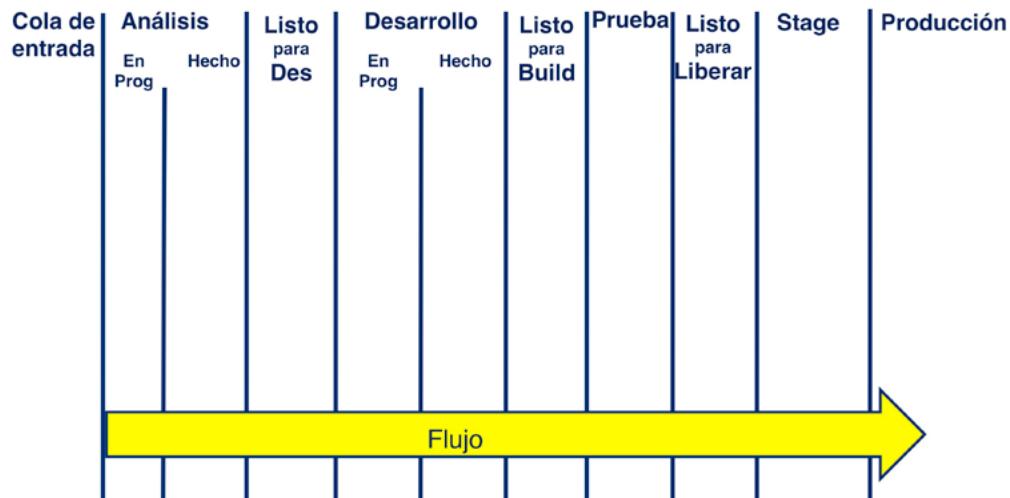


Figura 6.3 Flujo de trabajo con buffers y colas agregadas

Por último, añada las colas o buffers que considere necesarias. Existen diversas escuelas de pensamiento sobre esto y se trata en realidad de un tema avanzado. Una discusión completa sobre dónde poner los buffers y como dimensionarlos está más allá del alcance de este libro, por lo que considere suficiente la descripción de dos enfoques populares.

La primera escuela de pensamiento nos dice que no hay que cuestionar la ubicación del cuello de botella o la fuente de variabilidad que requerirá un buffer. En su lugar, implemente el sistema y espere a que el cuello de botella se revele a si mismo y entonces haga los cambios para introducir el buffer.

Una variante de esto sugiere que los límites del WIP deben ser determinados de manera muy holgada inicialmente tal que la variabilidad, el desperdicio y los cuellos de botella no tengan un impacto significativo en el sistema de arrastre cuando es implementado por primera vez. Esto se discute más de lleno en el capítulo 10 y más adelante en los capítulos 17 y 19.

Otra escuela de pensamiento toma un enfoque distinto. Sugiere que en lugar de implementar límites holgados de WIP para evitar una introducción retadora, cada estado debe tener buffer y los pasos de actividad deben tener límites estrictos. Los cuellos de botella y la variabilidad se revelarán por si mismos en base a que tan llenos estarán los buffers. Cambios pequeños y simples se pueden hacer para reducir los tamaños de los buffers y eventualmente podrá eliminar los buffers innecesarios.

Al momento de escribir este libro, no se tiene suficiente evidencia para sugerir cual enfoque es mejor.

Algunos equipos han adoptado la convención de mostrar los buffers y alinean las columnas utilizando una tarjeta girada 45 grados. Esto provee un indicador visual fuerte de cuanto trabajo está fluyendo en lugar de estar haciendo cola en cualquier instante. Esto le permite al equipo y otras partes interesadas “ver”, literalmente, el monto de costo económico (desperdicio) en el sistema.

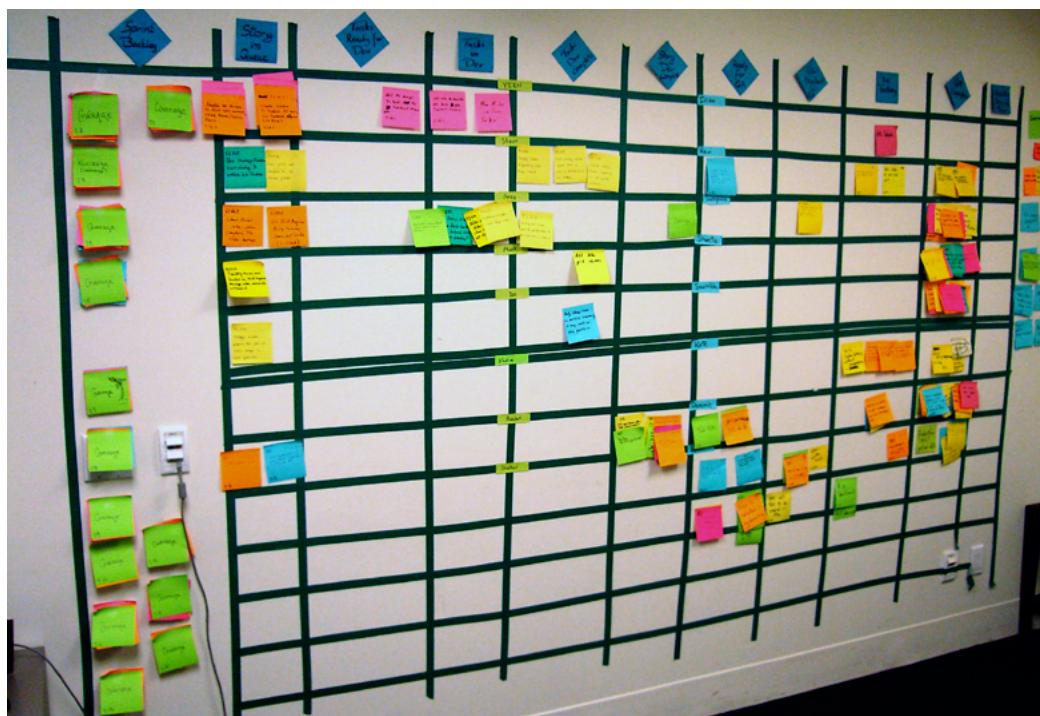


Figura 6.4 Pared de tarjetas ilustrando el uso de tarjetas en forma de diamante en el encabezado de las columnas de buffer o de cola
(Cortesía de Liquidnet Holdings, Inc.)

Análisis de demanda

Para cada tipo identificado de trabajo, usted debe hacer un estudio de su demanda. Si usted tiene datos históricos utilícelos para hacer un estudio cuantitativo. De lo contrario, un análisis subjetivo derivado de anécdotas será suficiente. Por ejemplo, en el ejemplo del XIT de Microsoft del capítulo 4, hay dos tipos de trabajo, Solicitudes de Cambio y Cambios de Texto en Producción. Podría argumentarse que las Solicitudes de Cambio deberían haberse separado en dos tipos, Defectos de Producción y Solicitudes de Cambio (para nueva funcionalidad). Si hoy yo fuera coach de este equipo hoy, recomendaría que hicieran seguimiento de cuatro tipos de trabajo en total: Solicitudes de Cambio, Defectos de Producción, Cambios de Texto en Producción y Bugs (es decir defectos no escapados a producciónn).

Estudiaríamos la demanda de cada uno de estos tipos. La demanda de los CTP llegaba en ráfagas. Podían no haber CTP durante seis semanas y luego en una sola semana habría una ráfaga de tal vez diez llegando casi simultáneamente. Los CTP eran pequeños y rápidos de implementar. Esto significa que su impacto no era grave. Diseñar un sistema que trate con tal demanda intermitente es difícil. Si los CTP representaran un esfuerzo importante, el sistema habría requerido contar con suficiente holgura con el fin de hacer frente a los CTP adecuadamente sin impactar la previsibilidad de Solicitudes de Cambio severamente.

Las Solicitudes de Cambio, por el contrario, llegaban a un ritmo mucho más constante. Mientras que su llegada era de naturaleza estocástica, la demanda era relativamente estable, con tal vez de cinco a siete nuevas solicitudes por semana. Sería posible graficar la tasa de llegada de los CTP en un gráfico y graficar la demanda para entender la tasa media de llegada y la difusión de la variación. El sistema kanban podría ser entonces diseñado con los recursos adecuados para hacer frente a esta demanda.

Algunos tipos de ítems de trabajo muestran demanda por temporada, tales como los requisitos reglamentarios. La nueva legislación fiscal afecta a los sistemas financieros y de nómina durante temporadas. En un caso en el que me encontré, el departamento de TI de un equipo de carreras de automóviles recibía los cambios de reglamentación de la organización regulatoria de ese deporte al comienzo de cada temporada de carreras. Podían recibir también algunos de los requisitos de regulación durante la temporada, pero el volumen durante el cierre de temporada era significativamente mayor, conforme los reglamentos principales de las carreras eran cambiados de un año al siguiente. Es importante comprender esta demanda de modo tal que el diseño del sistema kanban pueda ajustarse para hacer frente a la demanda de diferentes tipos de trabajo.

Asignando capacidad de acuerdo a la demanda

Una vez que tenga un entendimiento de la demanda, puede decidir cómo asignar la capacidad dentro del sistema kanban para hacer frente a esa demanda. El ejemplo en la Figura 6.5 muestra tres carriles de nado, uno para cada tipo de trabajo, a saber, las solicitudes de cambio; el trabajo de mantenimiento interno, como la refactorización de código y los cambios de texto en producción. La asignación es de 60% para las solicitudes de cambios, el 10% para el trabajo de refactorización de código y el 30% para los cambios de texto en producción. Teniendo en cuenta un análisis de la demanda que muestra que los cambios de texto en producción llegan en ráfagas, la asignación mostrada sugiere que holgura significativa está siendo reservada para tratar los cambios de texto en producción con urgencia en cuanto llegan sin afectar las fechas de vencimiento de otros trabajos. La asignación de capacidad debe ajustarse al perfil de riesgo. Si, por ejemplo, es aceptable que el rendimiento de fecha de vencimiento caiga cuando llegan cambios de texto en producción llegan y para los tiempos de entrega de las solicitudes de cambio ser más largos y menos predecibles, entonces la asignación podría ser diferente. Tal vez un 85% para las solicitudes de cambio, 10% para el mantenimiento y 5% para los cambios de texto en producción. Sin embargo, otra opción sería la de dejar un carril de nado para cambios de texto en producción, pero sin asignar ninguna capacidad para eso y adoptar una política de sobrepassa el límite del trabajo-en-progreso cuando una ráfaga de cambios de texto en producción llegue. Esta política eliminará holgura y producirá un resultado económico óptimo durante la operación normal. Sin embargo, cuando una ráfaga de cambios de texto en producción llegue, otros trabajos pueden verse gravemente afectados, tanto en el tiempo de entrega como en la previsibilidad. Esta fue la opción elegida en el ejemplo real del capítulo 4, no había capacidad adicional reservada para hacer frente a cambios de texto en producción.

Cola de entrada	Análisis		Listo para Des	Desarrollo		Listo para Build	Prueba	Listo para Liberar	...
	En Prog	Hecho		En Prog	Hecho				
Req. de Cambio 60%									
Mantenimiento 10%									
Cambio de Texto en Producción 30%									

Figura 6.5 Tablero de Kanban con carriles de nado para los tipos, indicando asignación de capacidad

Más adelante, cuando discutimos como poner los límites de trabajo-en-progreso, utilizaremos esa información de asignación para poner límites específicos para las colas en cada carril de nado.

Más adelante, cuando discutimos como poner los límites de trabajo-en-progreso, utilizaremos esa información de asignación para poner límites específicos para las colas en cada carril de nado.

Anatomía de una tarjeta de ítem de trabajo

Cada tarjeta visual que representa una pieza discreta de trabajo valioso para el cliente tiene varias piezas de información en él. El diseño de la tarjeta es importante. La información en las tarjetas debe facilitar el sistema de arrastre y facultar a los individuos para que tomen sus propias decisiones de toma. La información en un ticket puede variar por el tipo de ítem de trabajo o por clase de servicio (discutido en el capítulo 11).

En la Figura 6.6, el número de seguimiento electrónico se muestra en la esquina superior izquierda de la tarjeta adhesiva y es utilizado como el identificador único del ítem y ligarlo a la versión electrónica del sistema de seguimiento. El título del ítem está escrito en el medio. . La fecha en la que el ítem entró al sistema se muestra en la parte inferior izquierda y tiene doble propósito: facilita usar una cola tipo FIFO para la clase de servicio estándar y permite a los miembros del equipo ver cuantos días han transcurrido del SLA, o acuerdo de servicio (descrito en el capítulo 11). Para ítems de clase de servicio de fecha fija, la fecha de entrega requerida se escribe en la parte inferior derecha del ticket.

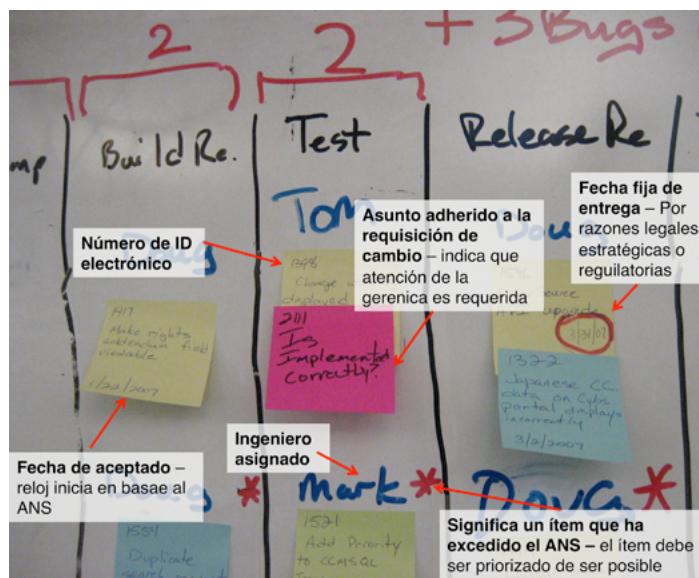


Figura 6.6 Acercamiento de la pared de tarjetas mostrando la anatomía de las tarjetas de ítems de trabajo

En el ejemplo mostrado en la figura 6.6, alguna otra información es mostrada afuera del ticket. Una estrella indica que el ítem está retrasado con respecto al tiempo de entrega límite especificado en el acuerdo de nivel de servicio. Más recientemente he visto lograrse esto poniendo una etiqueta adhesiva en la esquina superior derecha del ticket. El nombre de la persona asignada también se escribe arriba del ticket. Debido a que la persona asignada cambiará conforme el ticket fluye a lo largo del tablero, no es deseable escribir el nombre en el ticket. Sin embargo, implementaciones más recientes tienen pequeñas tarjetas de nombre adheridas a los ítems, magnetos (cuando el pizarrón es magnético) y tarjetas adhesivas o magnetos con avatares de los miembros del equipo. Personajes tales como los de *South Park* son populares como avatares. Cualquier mecanismo que le permite a los miembros del equipo y directores inmediatos saber de un vistazo quien está trabajando en qué es suficiente.

Como regla general, el diseño del ticket utilizado para representar una pieza individual de trabajo debe tener suficiente información para facilitar las decisiones de la administración del proyecto, tales como qué ítem tomar a continuación, sin la intervención o dirección de un director. La idea es facultar a los miembros del equipo con transparencia de proceso, metas y objetivos de proyecto, e información sobre riesgo. Conforme más descubra sobre las clases de servicio y los acuerdos de nivel de servicio, descubrirá que Kanban facilita un mecanismo poderoso de auto-organización y administración-de-riesgos. Igualmente, Kanban, faculta a los miembros del equipo para tomar sus propias decisiones de priorización y asignación de tiempo, muestra respeto por los individuos y una confianza en el sistema (o diseño de proceso). Una tarjeta de ítem de trabajo bien diseñada es un activador clave de una cultura de alta confianza y una organización Lean.

Seguimiento electrónico

El seguimiento electrónico ha sido una característica de los sistemas kanban en el desarrollo de software desde que fueron introducidos por primera vez en 2004. Sin embargo es opcional. Para los equipos distribuidos geográficamente, o aquellos que tienen políticas que permiten a los miembros del equipo trabajar desde sus hogares uno o más días por semana, el seguimiento electrónico es esencial. El seguimiento electrónico se puede realizar con sistemas básicos de gestión de tickets o de seguimiento de ítems de trabajo, tales como Jira, Microsoft Team Foundation Server, Fog Bugz y HP Quality Center. Sin embargo, un sistema más potente le permitirá visualizar el seguimiento de ítems de trabajo como si estuviera en una pared de tarjetas.

Al el momento de escribir estas líneas, una serie de herramientas basadas en Internet y basados en aplicaciones están surgiendo en el mercado para proporcionar seguimiento electrónico utilizando tableros visuales que simulan paredes de tarjetas con columnas, límites de WIP y otros aspectos esenciales de Kanban. Estos incluyen, pero no se limitan a:

Lean Kit Kanban, Agile Zen, Target Process, Silver Catalyst, RadTrack, Kanbanery, VersionOne, Greenhopper para Jira, Flow.io y algunos otros proyectos adicionales de opensource que añaden interfaces de Kanban a herramientas tales como Team Foundation Server y FogBugz. La Figura 6.7 muestra un ejemplo de AgileZen.

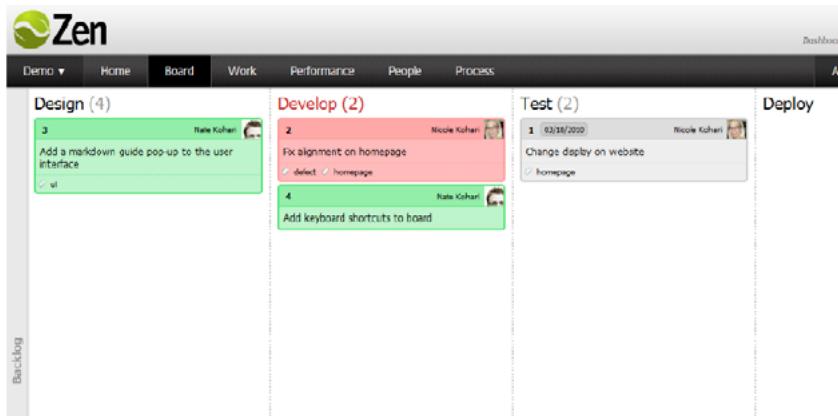


Figura 6.7 Pantalla de AgileZen como un ejemplo de una herramienta electrónica de seguimiento

El seguimiento electrónico es necesario para los equipos que aspiran a mayores niveles de madurez de la organización. Si usted anticipa la necesidad de la gestión cuantitativa, el rendimiento del proceso organizacional (comparando la eficiencia entre los sistemas kanban, los equipos o los proyectos) y/o el análisis causal y la resolución (análisis de causa raíz a partir de datos estadísticamente fiables), usted deseará usar una herramienta electrónica desde el principio.

Estableciendo los límites de entrada y salida

Alinee el diseño del sistema kanban y la pared de tarjetas con la decisión tomada antes de limitar el límite de control de WIP. Es probable que los socios de niveles anteriores y posteriores del proceso pidan más tarde visualizar su trabajo en su pared de tarjetas. Sin embargo, es mejor proveer transparencia en su propio trabajo primero y esperar a que otros le pidan ser parte de su iniciativa de Kanban.

En el ejemplo que se muestra en la Figura 6.8, la cola de entrada está designada como "E.R." para indicar que está listo para ingeniería (en inglés: *Engineering Ready*). Tiene sentido establecer el punto de entrada en este paso dentro del ciclo de vida porque el departamento de análisis de negocios en el nivel anterior en el proceso reporta hacia arriba a través de una parte diferente de la estructura de la organización. Había poca confianza o colaboración entre los directores de los dos grupos. La cola de entrada estaba, por lo tanto, nutrida a partir del *backlog* de requerimientos generados por el departamento de análisis de negocios.

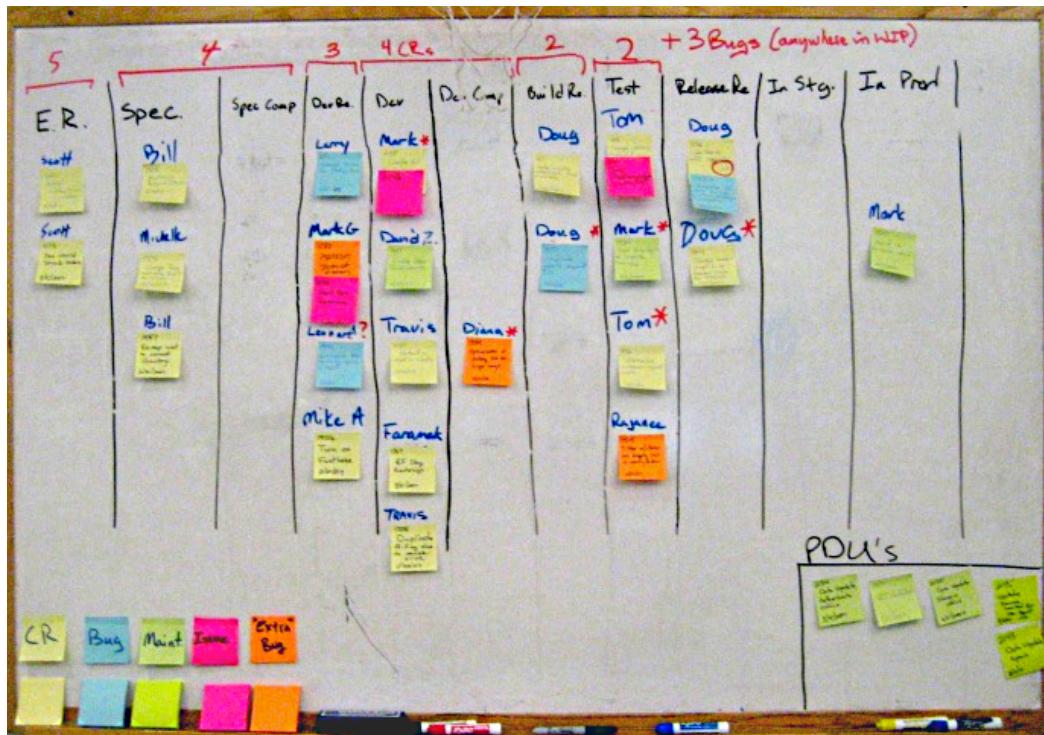


Figura 6.8 Mostrando la Cola de Entrada “Listo para Ingeniería” (E.R.)

En este ejemplo, la entrega al nivel posterior es el despliegue a producción. Una vez que el software es desplegado y entregado al departamento de operaciones de sistemas y redes paralelo mantenimiento diario y el apoyo, se considera fuera del ámbito.

Haciendo frente a la concurrencia

Una ocurrencia común al diseñar una pared de tarjetas para un sistema kanban es un proceso en el que dos o más actividades pueden ocurrir al mismo tiempo, por ejemplo, desarrollo de software y desarrollo de pruebas. Hay dos patrones básicos para hacer frente a esta situación. Uno no modelarlo para nada; tan solo deje una sola columna en la cual ambas actividades pueden ocurrir reuniones (Figura 6.9). Esto es simple, pero muchos equipos no lo han preferido. Algunos equipos se han adaptado este modelo mediante el uso de diferentes colores o formas de tickets para mostrar las diferentes actividades.

La otra opción es dividir el tablero verticalmente en dos (o más) secciones (ver Figura 6.10).

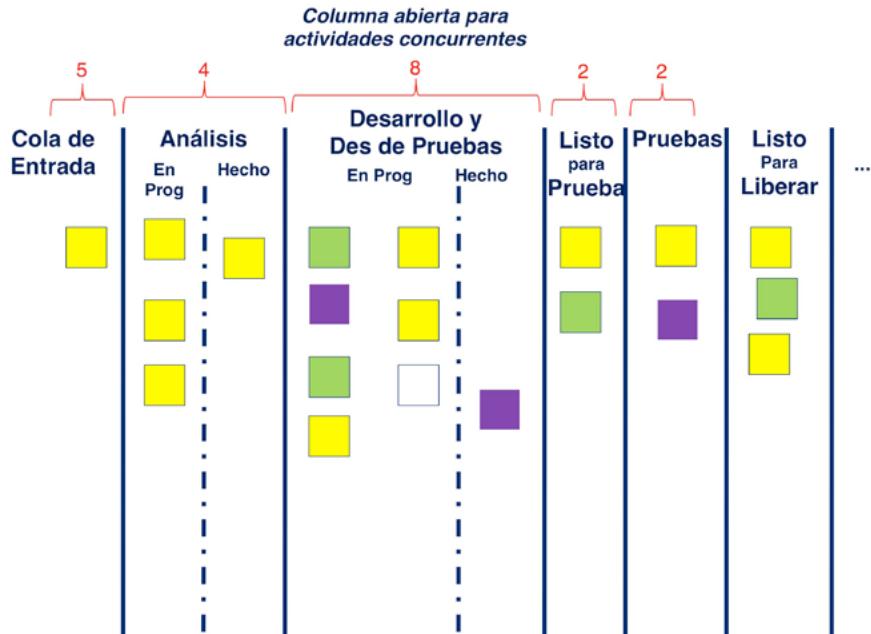


Figura 6.9 Columna abierta para actividades concurrentes

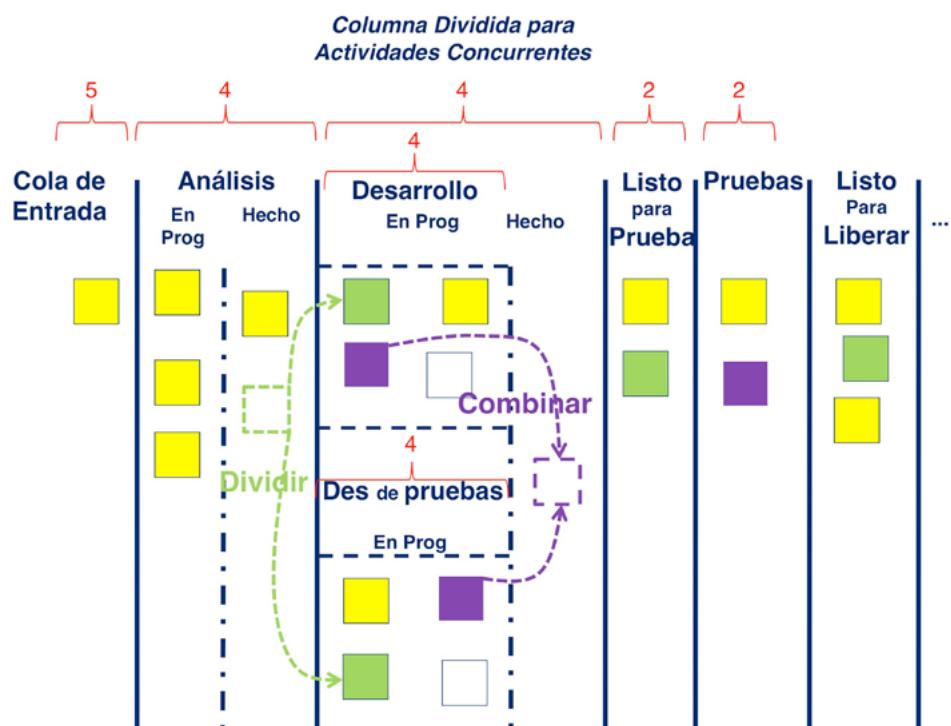


Figura 6.10 Columna dividida para actividades concurrentes

En este ejemplo, es necesario tener un mecanismo de etiquetado para atar los ítems de la parte superior e inferior del tablero. Esto puede implicar simplemente el uso de, por ejemplo, la esquina superior derecha del ticket para hacer referencia cruzada al ítem asociado. En un buen sistema de seguimiento electrónico es posible vincular los ítems relacionados, tales como actividades de desarrollo y prueba.

Haciendo frente a actividades desordenadas

Particularmente en trabajo altamente innovador y experimental, puede haber varias actividades que tienen que suceder con parte del trabajo valorado por el cliente, pero estas actividades no tienen que suceder en ningún orden en particular. En estas circunstancias, es importante darse cuenta de que Kanban no debe obligarlo a completar las actividades en un orden determinado. Lo que es más importante al modelar su sistema kanban es que debe reflejar la manera en que el trabajo real se hace.

Hay un par de estrategias para el problema de las actividades múltiples desordenadas. La primera es similar a cuando hacemos frente a la concurrencia: Simplemente tenga una columna como un cubo para las actividades y no rastree de forma explícita en el tablero cuál de ellas se ha completado.

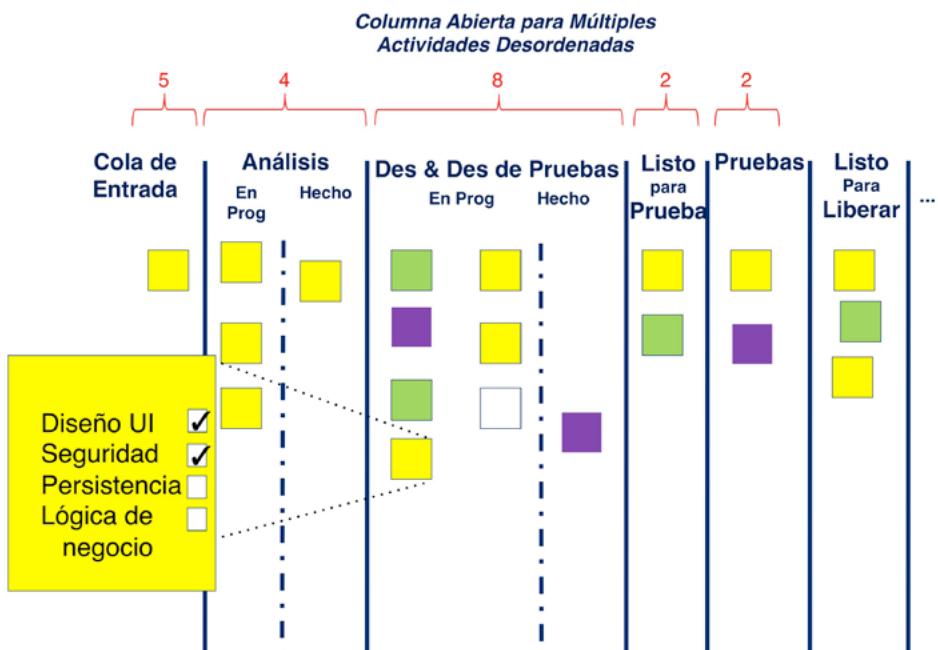


Figura 6.11 Columna abierta para múltiples actividades desordenadas

La segunda y potencialmente más poderosa elección, es modelar las actividades de una manera similar a las actividades concurrentes. En este diseño, como se muestra en la Figura 6.11, los tickets tienen que moverse verticalmente hacia arriba y hacia abajo de la columna conforme son arrastrados dentro de cada una de las actividades específicas. Visualizando que actividades han sido completadas en cada ítem puede hacerse modificando el diseño del ticket para que tenga una pequeña caja para cada actividad. Cuando la actividad se ha completado, la caja puede ser llenada para señalizar visualmente que el ítem está listo para ser llevado a otra actividad en la misma columna. Si todas las cajas están llenas, el ítem está listo para ser llevado a la siguiente columna en el tablero, o puede ser trasladado a la columna de “hecho” (en inglés: *done*) (ver Figura 6.12).

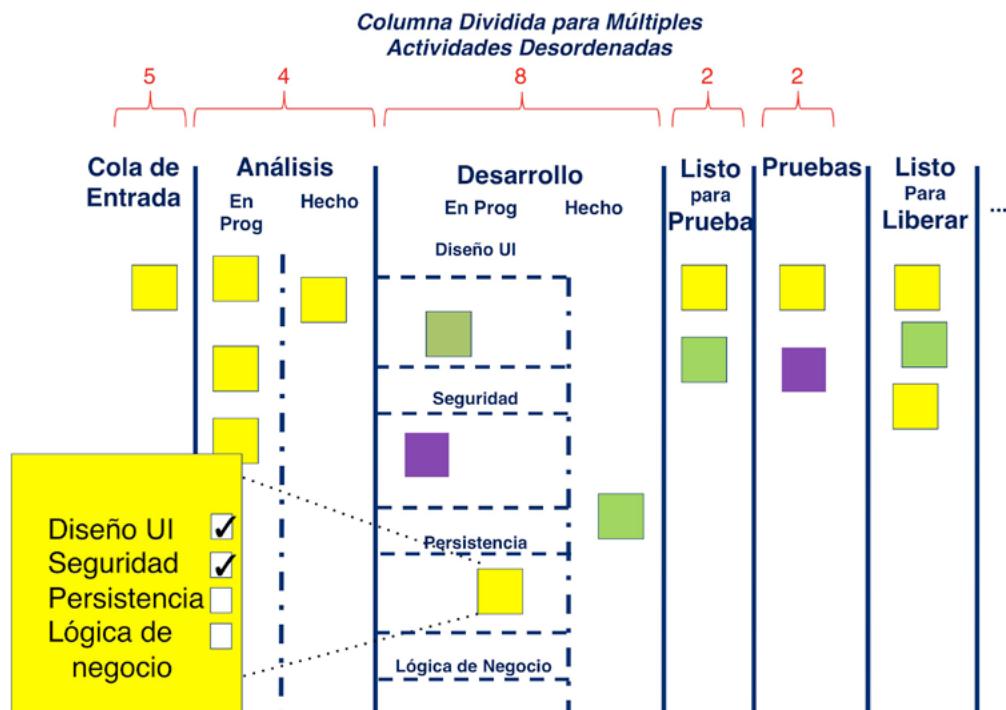


Figura 6.12 Columna dividida para múltiples actividades desordenadas

Para llevar

- ❖ Decida los límites externos del sistema kanban. A menudo es mejor limitarlo al rango de medición inmediata de control político. No fuerce la visualización, la transparencia y los límites de WIP en cualquier departamento que no se ofrezca voluntariamente a colaborar.
- ❖ Modele la pared de tarjetas para alinearse con las decisiones de límite hechas en relación con el límite del WIP y la visualización de trabajo.
- ❖ Defina los tipos de ítems de trabajo y modele como su trabajo fluye. Algunos tipos pueden no requerir de cada paso en el flujo.
- ❖ Diseñe la tarjeta del ítem de trabajo para que tenga la suficiente información para facilitar la auto-organización para el arrastre y para permitir a los miembros del equipo que tomen decisiones de buena calidad con respecto al riesgo basados en el tipo de ítem de trabajo, acuerdos de nivel de servicio y clases de servicio.
- ❖ Utilice una herramienta electrónica de seguimiento si el equipo está distribuido, tiene alguna política de trabajo-desde-casa, o aspira a comportamientos de más alta madurez que requieran la información cuantitativa que un sistema electrónico puede proveer.
- ❖ Donde sea apropiado, discuta métodos para gestionar concurrencia en las actividades y elija como modelarlos y visualizarlos.
- ❖ Donde sea apropiado, discuta métodos para gestionar actividades que no necesitan seguir un flujo específicamente ordenado y elija como modelarlos y visualizarlos.

❖ CAPÍTULO 7 ❖

Coordinación con sistemas Kanban

Control visual y arrastre

Cuando la gente habla sobre Kanban, la forma más popular de coordinación que viene a la mente es la pared de tarjetas. Por lo general, los límites del trabajo-en-progreso se dibujan en el tablero en la parte superior de cada columna o a través de varias columnas. El arrastre es señalado si el número de tarjetas en una columna es menor que el límite indicado. En la figura 7.1, podemos ver que el límite indicado arriba de la columna de análisis es de 4 ítems. Sin embargo, hay tan sólo tres tarjetas en esa columna. Porque $4 - 3 = 1$, esto indica que se puede arrastrar un ítem en la columna *Spec* (la función de análisis-de-sistemas) a partir de la cola de entrada Listo para Ingeniería (en inglés: *Engineering Ready—E.R.*). A su vez, la cola de entrada tiene un límite de cuatro. Actualmente quedan tan sólo dos ítems en esa cola. Cuando arrastramos uno en *Spec*, sólo quedará pendiente un ítem ($5 - 1 = 4$). Esto indica que podemos priorizar cuatro ítems nuevos en la cola de entrada durante la siguiente reunión de priorización.

Cuando el equipo decide tomar un ítem, pueden escoger cual ítem tomar basados en la información visual disponible, tal como el tipo de ítem de trabajo, la clase de servicio, la fecha límite (cuando es aplicable) y la antigüedad del ítem de trabajo. Las políticas de arrastre relacionadas con la clase de servicio son discutidas en el capítulo 11.

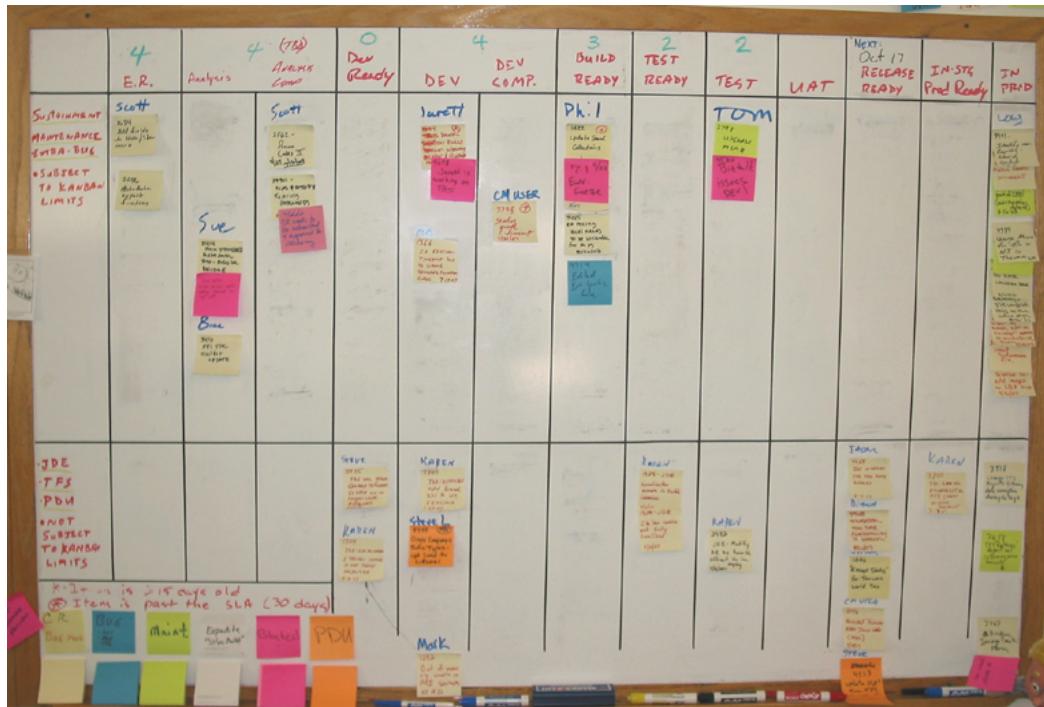


Figura 7.1 Mostrando límites de Kanban arriba de las columnas en una pared de tarjetas

La figura 7.2 muestra un acercamiento de las notas adhesivas que representan los ítems de trabajo en nuestra pared de tarjetas. El color es utilizado para comunicar una combinación del tipo de ítem de trabajo y la clase de servicio. El nombre del propietario o miembro del personal asignado está escrito arriba de la tarjeta. A algunos equipos les gusta utilizar pequeñas notas adhesivas adicionales con nombres o avatares pegados al ítem de trabajo para indicar quién está trabajando en cada uno de ellos. Esto permite que todos en el equipo vean quién está trabajando en cada cosa.

El número de seguimiento electrónico se muestra en la esquina superior derecha de la tarjeta. La fecha en la que un ítem entró en la cola de entrada se muestra en la parte inferior izquierda. La antigüedad del ítem se puede deducir a partir de esta fecha. Si un artículo es de una clase de servicio que tiene una fecha de entrega garantizada, entonces eso se muestra en la parte inferior derecha. Si un ítem está retrasado, se señala con una estrella roja arriba de la tarjeta en la parte superior derecha. Si algo está bloqueado, un ticket de color rosa se pone en el ítem bloqueado. En el ejemplo de la Figura 7.2, el asunto es con un ítem de trabajo de primera clase y tiene por ende su propio número de seguimiento electrónico; la fecha de cuándo entró en el sistema y, potencialmente, un miembro del personal asignado.

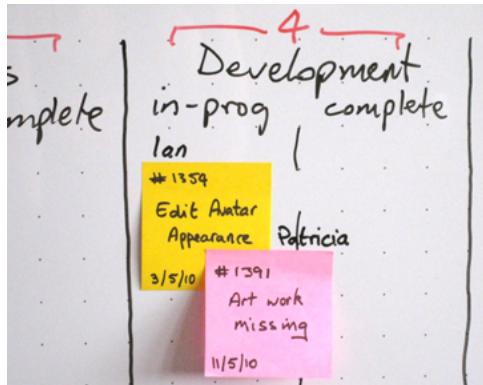


Figura 7.2 Acercamiento de la pared de tarjetas mostrando una tarjeta de asunto adherida al ítem bloqueado

Este esquema es idiosincrático a la primera implementación de kanban en Corbis. Su implementación será, casi seguramente, diferente. Sin embargo, es probable que usted desee mostrar visualmente al miembro del personal asignado, la fecha de inicio, el número de seguimiento electrónico, el tipo de ítem de trabajo, la clase de servicio y alguna información sobre el estado, tal como si el artículo está retrasado. La meta es comunicar visualmente la información suficiente para hacer que el sistema sea auto-organizado y auto-acelerado a nivel de equipo. Como mecanismo de control visual, el tablero de Kanban debe permitir a los miembros del equipo tomar trabajo sin la asistencia de su director.

Seguimiento electrónico

Como alternativa o como suplemento a una pared de tarjetas, un sistema electrónico se utiliza a menudo para el seguimiento del trabajo en un sistema kanban. Algunas de las herramientas disponibles para ese fin se enumeran en el capítulo 6. Para una lista más actualizada consulte el sitio web de la Limited WIP Society, <http://www.limitedwipsociety.org/>.

Con mi equipo, hemos implementado nuestra propia aplicación, llamada *Digital Whiteboard* (ver Figura 7.3), sobre de *Team Foundation Server*. En el caso de estudio en el capítulo 4, el seguimiento electrónico se realizó utilizando una herramienta interna de Microsoft llamada *Product Studio*. Este fue un precursor de *Team Foundation Server* y desde 2005 Microsoft ha estado utilizando *Team Foundation Server* para su propio control interno de proyectos de desarrollo.

La aplicación mostrada en la Figura 7.3 muestra los límites de kanban agrupados sobre las columnas. Es capaz de mostrar visualmente cuando el límite de kanban se han excedido. También muestra el número de ítems de estado para cada ítem de trabajo, incluyendo los diferentes iconos que muestran si un ítem se atrasa o está bloqueado con un problema.

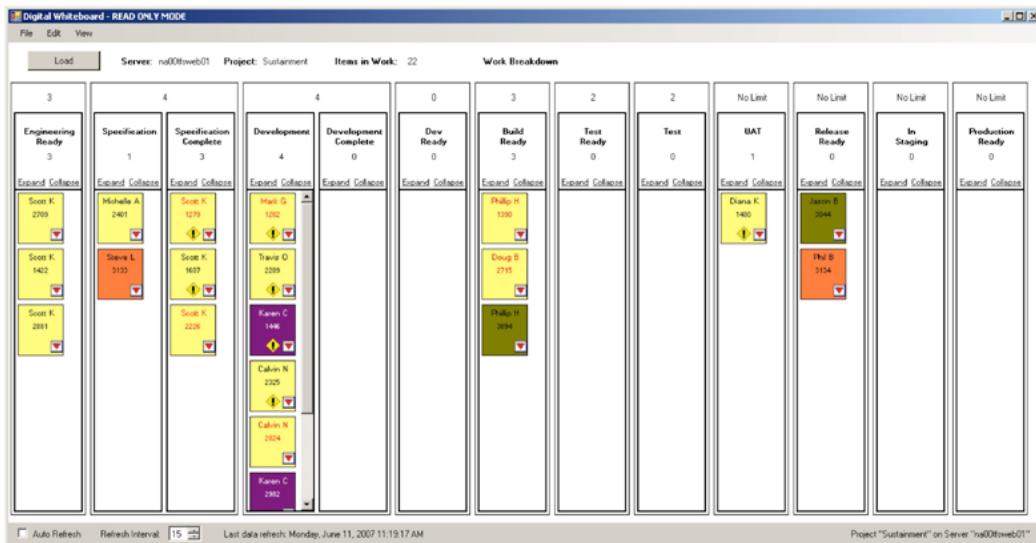


Figura 7.3 Pizarrón Blanco Digital utilizado en Corbis

El seguimiento electrónico es importante para los sistemas kanban, ya que permite varias cosas que no son viables con una pared de tarjetas simple. El seguimiento electrónico permite la recopilación de datos que se pueden utilizar para generar métricas y reportes tanto para la gestión del día a día como para un uso retrospectivo, por ejemplo, en las revisiones mensuales de las operaciones.

Reuniones de pie diarias

Las reuniones de pie son un elemento común de los procesos de desarrollo ágil. Normalmente llevan a cabo en la mañana antes de comenzar a trabajar y tienen un formato acordado. Una reunión de pie típica es para un solo equipo de hasta doce personas—normalmente cerca de seis. El formato generalmente consiste en que el grupo opera alrededor de tres preguntas: ¿Qué hiciste ayer? ¿Qué vas a hacer hoy? ¿Estás bloqueado o necesitas ayuda? Cada miembro del equipo responde a estas preguntas y entonces el equipo se coordina para hacer su trabajo del día.

Las reuniones de pie han evolucionado de manera diferente en Kanban. La necesidad de pasar por todo el equipo con las tres preguntas se evita mediante la pared de tarjetas. La pared contiene toda la información sobre quién está trabajando en qué. Los asistentes que acuden regularmente pueden ver qué ha cambiado desde ayer y si algo está bloqueado o no es visualmente evidente. Por lo tanto, las reuniones de pie tienen un formato diferente con un sistema kanban. El enfoque se centra en el flujo de trabajo. El facilitador, normalmente un director de proyecto o de un director de línea, “caminará el tablero”. La convención

se ha desarrollado para trabajar hacia atrás—de derecha a izquierda (en la dirección de arrastre)—a través de los tickets en el tablero. El facilitador puede solicitar una actualización sobre el estado de un ticket o simplemente preguntar si hay alguna información adicional que no está en el tablero y que puede ser desconocida por el equipo.

Debe ponerse particular énfasis en ítems que están bloqueados (disponer de un ticket rosa adjunto) o retrasados debido a defectos (tiene una serie de tickets azules adjuntos). Se pueden hacer preguntas sobre ítems que parecen estar atascado y no se han movido en algunos días. Algunos equipos han ideado formas para visualizar esto. Por ejemplo, en una manufacturera italiana de automóviles deportivos y su equipo de carreras de vehículos, marcan un punto al un lado del ticket por cada día que el ticket se queda en un solo lugar. Esto permite al equipo preguntarse si un ítem debe ser marcado como bloqueado si no fluye de manera activa. Esto mejora la capacidad de gestión de problemas de la organización (descrita con más detalle en el capítulo 20). El equipo discutirá brevemente quien está trabajando en un problema y cuando será resuelto. También habrá una convocatoria para cualquier otro elemento de bloqueo que no esté en el tablero y para que cualquier persona que necesite ayuda hable. Equipos más avanzados y maduros no necesitarán recorrer todas y cada una de las tarjetas en la pared. Ellos tienden a centrarse sólo en los tickets que están bloqueados o tienen defectos. Este mecanismo permite que las reuniones de pie se expandan a un número mucho mayor de personas; Daniel Vacanti gestionó exitosamente reuniones de pie con más de 50 personas en un proyecto de Corbis en 2007 donde, a pesar del gran tamaño del equipo, la reunión tomaba tan solo unos 10 minutos cada mañana.

La pos-reunión

La pos-reunión consiste en enyuntar a pequeños grupos de 2 ó 3 personas. Esto surgió como una conducta espontánea porque los miembros del equipo quería discutir algo que tenían en mente: tal vez un problema de bloqueo, tal vez un problema de diseño o arquitectura técnica, pero más a menudo debido a una cuestión relacionada con el proceso. La pos-reunión es un elemento vital de la transformación cultural que emerge con Kanban. Las posreuniones generan ideas de mejora y resultan en un proceso personalizado y en innovación.

En proyectos más grandes, algunas pos-reuniones toman la forma de reuniones de pie estilo Scrum. Los equipos de hasta seis personas que trabajan juntos en una característica, historia, o requisito se reúnen brevemente para coordinar sus esfuerzos del día. Hay una diferencia interesante entre este comportamiento del proceso emergente de Kanban y Scrum. Con Scrum, los equipos se reúnen primero y luego envían a un representante a un Scrum de Scrums para coordinar un programa o un proyecto grande. En Kanban el comportamiento es invertido—la reunión a nivel de programa ocurre primero.

Reuniones de Reposición de Cola

Las reuniones para aprovisionar la cola sirven al propósito de priorizar en Kanban. Esta priorización se dice que es aplazados hasta el último momento razonable debido a la naturaleza del mecanismo de reposición de la cola y la cadencia de las reuniones. Las reuniones de aprovisionamiento de la cola se llevan a cabo con un grupo de representantes de negocio o propietarios de productos (siguiendo el uso popular de desarrollo Ágil). Se recomienda que estas reuniones se efectúen a intervalos regulares. Tener una cadencia constante para aprovisionar la cola reduce el costo de la coordinación para a cabo la reunión y ofrece seguridad y fiabilidad entre el equipo de negocio y el equipo de desarrollo de software.

El propósito de dicha reunión es llenar la cola de entrada del sistema kanban para una cadena de valor único, sistema o proyecto. Las partes interesadas que tienen un interés en las entregas del equipo y que tienen ítems esperando en el *backlog* deberán asistir a esta reunión. Los asistentes de parte de negocios debe ser lo más senior posible dentro de sus organizaciones. Personas en puestos más senior pueden tomar más decisiones y con frecuencia tienen acceso a la información contextual más amplia. Esto mejora la calidad de la toma de decisiones y optimiza el proceso de selección para aprovisionar la cola.

Idealmente, una reunión de priorización implicará varios propietarios de producto o de gente de negocios de los grupos potencialmente competidores dentro de la empresa. La tensión creada por esto en realidad se convierte en una influencia positiva para la buena toma de decisiones y estimula un ambiente sano y colaborativo con el equipo de desarrollo de software. Si tan sólo un propietario de un producto atiende, existe la posibilidad de que la interacción sea adversaria.

Otras partes interesadas deben estar presentes en la reunión, idealmente incluyendo: cualquier persona responsable de entregas, por ejemplo, un director de proyecto; por lo menos un director técnico funcional, tales como un director de desarrollo o pruebas, o un director técnico más senior; algunas personas que pueden evaluar los riesgos técnicos, por ejemplo, un arquitecto técnico o de datos; un experto en usabilidad; un especialista en operaciones y sistemas; y un analista de negocios. Con mi equipo en 2007, un director de desarrollo, el director del equipo de análisis y, ocasionalmente, el arquitecto de la empresa o el arquitecto de datos asistieron a la reunión. Los directores de desarrollo se turnaban, asistiendo a la reunión en un calendario rotativo.

La cadencia de las reuniones de priorización afectará la forma en que la cola es arreglada en el sistema kanban y por lo tanto afectará el tiempo de entrega global a través del sistema. Para maximizar la agilidad del equipo, se recomienda que las reuniones sean tan frecuentes como sea razonablemente posible, una frecuencia recomendable es semanal.

Algunos equipos han evolucionado en ser dirigidos por prioridades en lugar de utilizar una reunión ordinaria. Esto se recomienda sólo para organizaciones más maduras en las que todos los interesados que participan en las reuniones están disponibles bajo demanda.

En el caso de estudio de Microsoft en el capítulo 4, el director de proyecto creó disparadores de base de datos que le alertaban en cuanto había un espacio libre en la cola de entrada. Iniciaba entonces una discusión depriorización con los cuatro propietarios de producto vía correo electrónico, alertándoles qua habría un espacio disponible para priorización. Una discusión electrónica se llevaba a cabo y un nuevo ítem del *backlog* era seleccionado. Este proceso generalmente toma aproximadamente dos horas. Al tener este sistema por demanda en lugar de la reunión semanal, el tamaño de la cola de entrada pudo ser reducida, lo que condujo a una posteriores mejoras del tiempo de entrega a través del sistema.

Reuniones de planificación de entrega

Las reuniones de planificación de entrega se llevan a cabo para planificaren entregas a los niveles posteriores de la organización. Si las entregasocurren regularmente, con una cadencia de, digamos, cada dos semanas, entonces tiene sentido programar la actividad de planificación de entrega para que se realice periódicamente. Esto reduce el costo de la coordinación para llevar a cabo la reunión y asegura que todos los que necesitan para asistir tendrá el tiempo disponible.

La persona responsable de coordinar la entrega, por lo general un director de proyecto, normalmente conduce las reuniones de planificación de la entrega. Cualquier otra parte interesada bebe ser invitada: Normalmente incluye a especialistas en gestión de la configuración, especialistas en la operación de sistemas y redes, desarrolladores, ingenieros de pruebas, analistas de negocio y para todos estos contribuyentes individuales, sus supervisores inmediatos o directores. Los especialistas están presentes debido a sus conocimientos técnicos y capacidades de evaluación de riesgos. Los directores están presentes se puedan tomar decisiones.

Una organización madura tendrá una lista de control o un marco para la entrega que facilite la planificación. Algunas de las cosas que deben considerarse son los siguientes.

- ¿Qué ítems del sistema están (o estarán) listos para su entrega?
- ¿Que se necesita para de hecho liberar cada ítem en producción?
- ¿Qué pruebas se requieren después de la entrega para validar la integridad de los sistemas de producción?
- ¿Qué riesgos están implicados?
- ¿Cómo se están mitigando esos riesgos?
- ¿Qué planes de contingencia son necesarios?

- ¿Quién debe participar en entrega y estar presente durante el empuje a producción (u otro mecanismo de entrega)?
- ¿Cuánto tiempo toma la entrega?
- ¿Qué otras logísticas estarán involucradas?

El resultado debe ser una plantilla completa que represente un plan de entrega. Con los equipos que son particularmente sofisticados, he visto el guión del plan de entrega como una orquestación de los procedimientos a ser ejecutados en un orden determinado.

En una reunión grande, puede que no sea posible completar el plan de entrega y algún trabajo independiente de seguimiento puede requerirse por parte del director de proyecto.

Triaje

Triaje es un término tomado de la profesión médica, se refiere a la práctica de evaluar y clasificar en categorías a los pacientes de emergencia para asignarles prioridad de atención. El sistema fue utilizado por primera vez en las unidades médicas de campos de batalla, donde los pacientes eran separados en tres categorías: más allá de poder ser ayudado y propenso a morir pronto, probable que sobreviva únicamente si se le da atención médica y probable que sobreviva sin tratamiento inmediato. Las salas de emergencia ahora utilizan un sistema similar para dar prioridad a los pacientes a medida que llegan para recibir tratamiento.

Triaje se adoptó en el desarrollo de software para la clasificación de defectos (*bugs*) durante la fase de estabilización de un proyecto de software tradicional. Triaje se utiliza para clasificar y priorizar los defectos que serán corregidos, a diferencia a los errores que no serán corregidos y que se les permitirá escapar a producción cuando el producto es lanzado. Un triaje habitual involucra a un líder de pruebas, un supervisor o director de pruebas, un líder de desarrollo, un supervisor o director de desarrollo y un propietario de producto.

Aún con Kanban tiene sentido hacer triaje de los defectos. Sin embargo, la aplicación más útil de triaje es sobre el *backlog* de los ítems que están esperando entrar en el sistema.

El triaje del *backlog* debe efectuarse a intervalos relativamente infrecuentes. (Nota: algunos métodos de desarrollo de software Ágil se refieren a esto como “aseando el *backlog*”). Mensual, trimestral y dos veces al año son intervalos populares entre los equipos. Los asistentes al triaje del *backlog* normalmente serán los propietarios del producto mismos o los representantes de negocio que asisten a la reunión de reposición de cola, junto con el director del proyecto. Los técnicos no suelen asistir en gran número. Tal vez un director técnico de nivel medio estará presente.

El propósito de un triaje de *backlog* es ir a través de cada ítem en el *backlog* y decidir si debe permanecer en el *backlog* o suprimirse. No es apilar rangos o establecer prioridades más allá de la simple elección de mantener-o-eliminar.

Algunos equipos han evitado la necesidad de triaje mediante automatización y política. El equipo de XIT de Microsoft del caso de estudio en el capítulo 4 se descartaba cualquier ítem de más de seis meses a un intervalo mensual regular. El razonamiento era que si un ítem no había sido seleccionado para la cola de entrada en seis meses, era poco probable que tuviera un valor significativo y por lo tanto poco probable que llegara a ser seleccionado. Si esto cambiaba era probable que fuera solicitado de nuevo y por lo tanto nada se perdería al ser eliminado del *backlog*.

El propósito de hacer el triaje del *backlog* es reducir su tamaño. El beneficio de tener un *backlog* pequeño es que facilita el debate de prioridades. Si hay 200 elementos en un *backlog*, tomará un monto de tiempo significativamente menor elegir a los ganadores en una reunión de priorización que si se tuvieran más de 2,000 elementos en el *backlog*.

Una buena regla de dedo podría ser que si el *backlog* supera el equivalente a tres meses de trabajo, es decir, el throughput de tres meses de entrega y todos los elementos en el *backlog* no pueden entrar al sistema dentro de ese monto de tiempo, será una buena idea podar el *backlog*. El tamaño adecuado del *backlog* variará en función de los distintos mercados y ámbitos. Los dominios con una alta volatilidad van a querer un *backlog* con capacidad de ítems para probablemente un mes de trabajo. Los dominios con muy baja volatilidad podrían tener un *backlog* con un máximo de un año de ítems.

Por lo tanto, existe una relación entre el tamaño del *backlog*, la volatilidad del dominio en que el sistema individual de kanban está en funcionamiento y la velocidad de entrega—o throughput—del equipo. Si un equipo entrega 20 historias de usuario al mes y el dominio tiene alguna, pero no excesiva, volatilidad, de modo tal que un *backlog* de tres meses es deseable, el *backlog* puede tener cerca de 60 ítems.

Revisión y escalamiento del registro de asuntos

Cuando los ítems de trabajo en el sistema kanban tienen impedimentos, se marcan como tal y se crea un ítem de trabajo sobre ese asunto. El asunto se mantendrá abierto hasta que el impedimento sea eliminado y el ítem de trabajo original pueda progresar a través del sistema. Revisar los asuntos pendientes, por lo tanto, es vital para mejorar el flujo a través del sistema.

La revisión del registro de asuntos debe ocurrir con frecuencia y regularidad. De nuevo, una cadencia regular reduce los costos de coordinación y asegura que las partes interesadas tengan tiempo para asistir. Organizaciones muy maduras pueden ser capaces de prescindir de las reuniones regulares y llevar a cabo reuniones bajo demanda. Esto podría ser adecuado si hay un número relativamente pequeño de asuntos que ocurren y si el incremento del costo de coordinación para efectuar reuniones bajo demanda es menor que el costo de llevar a cabo la reunión regularmente.

Las revisiones del registro de asuntos deben involucrar al director de proyecto y a quienes registraron los ítems como bloqueados. Las principales preguntas a responder son: “¿Quién está asignado al asunto y trabajando en él?” y “¿Cuándo se espera que esté resuelto?” Asuntos que no están progresando y están bloqueados en sí mismos sin poder avanzar deben ser escalados a nivel directivo.

Puede que no sea necesario tener presente a los directores en una revisión del registro de asuntos, pero es importante tener claramente definidas las rutas de escalamiento y las políticas. Cuando un asunto está bloqueado, el director de proyecto debe asumir la responsabilidad y escalar el asunto adecuadamente.

La gestión y escalamiento de asuntos es hecha habitualmente de una forma muy mala, aun en organizaciones de desarrollo ágil. Resolver problemas con rapidez, en particular los asunto externos al equipo de desarrollo, tales como la disponibilidad del entorno de desarrollo, requerimientos ambiguos, o la falta de equipo de prueba, mejora el flujo y mejora notablemente tanto la productividad del equipo como el valor entregado. La gestión y escalamiento de asuntos son disciplinas básicas que proporcionan un gran retorno. Su mejora debe ser una prioridad incluso para los equipos más inmaduros. Esto es discutido ampliamente en el capítulo 20.

Sticky friend

El concepto de *Sticky friend* se introdujo en Corbis para resolver un problema de coordinación. Se tenía una política que permitía el teletrabajo al menos un día por semana, especialmente para los empleados que vivían más lejos de la ciudad que la mayoría. La política se remonta varios años antes cuando se mudaron las oficinas de Bellevue a Seattle, Washington. El personal que hace teletrabajo es capaz de acceder al sistema de seguimiento electrónico, el control de versiones, los entornos de desarrollo y así sucesivamente a través de VPN. Ellos eran capaces de ver el trabajo que se les asignaba, trabajar en ello, completarlo y probarlo. Ellos eran capaces de actualizar el estado electrónico del trabajo, marcándolo como completado y disponible para ser tomado por niveles posteriores del proceso. Sin embargo, no podían mover el ticket en la pared de tarjetas por no estar físicamente presentes en la oficina.

La solución a esto fue que cada persona hiciera un acuerdo de igual a igual con alguien que estuviese presente en la oficina para actuar como su delegado. Cuando un trabajador a distancia completaba un ítem y cambiaba el estado electrónico, se ponía en contacto con su Sticky friend mediante mensajes instantáneos, correo electrónico o teléfono y le pedía que actualizara el tablero físico.

Los Sticky friends también facilitaban el desarrollo distribuido a través de distintas localidades geográficas. Esto era particularmente importante en Corbis, pues el grupo de

pruebas estaba en Chennai, India y tenía algunos desarrolladores especialistas en sistemas financieros en el Sur de California.

Sincronización a través de localidades geográficas

La sincronización de los equipos utilizando sistemas kanban a través de múltiples localidades geográficas surge una y otra vez como una cuestión elemental de quienes están considerando implementar un sistema kanban. A menudo, la persona supone que las primeras implementaciones de Kanban se realizaron en una sola ubicación geográfica y que yo (y otros defensores en los inicios de Kanban) no había considerado las dificultades de coordinación entre los equipos distribuidos geográficamente.

En realidad, lo opuesto es verdad. El primer equipo en Microsoft, en el capítulo 4, se encontraba ubicado en Hyderabad, India, con la dirección y los propietarios de producto localizados en Redmond, Washington. Corbis, descrito en el capítulo 5, también tenía personas en la India y otros lugares fuera de Seattle, tales como Los Ángeles y Nueva York, en adición a quienes hacían teletrabajo.

La clave para la coordinación a través de múltiples localidades es utilizar un sistema electrónico. Contar con solo una pared de tarjetas no es suficiente.

Además de seguimiento electrónico, será necesario mantener las paredes físicas de tarjetas sincronizadas, por lo menos, de manera diaria. Es importante asignar a alguien para asumir la responsabilidad de esta actividad en cada localidad. Un equipo con el que trabajamos en 2008 estaba distribuido entre Nueva York y Los Ángeles. Mantuvieron paredes de tarjetas (casi) idénticas en cada localidad e hicieron a un miembro del equipo responsable de mantenerlos sincronizados todos los días.

Algunos equipos inclusive coordinan reuniones de pie por teléfono o utilizando sistemas de video-conferencia. Previo a la reunión de pie, video conferencia, o llamada telefónica, el responsable local debía dedicar el tiempo necesario para asegurarse que el tablero electrónico estaba sincronizado con el sistema electrónico.

Para llevar

- ❖ La mejor práctica es utilizar tanto un muro físico de tarjetas como un sistema de seguimiento electrónico.
- ❖ Kanban es posible a través de múltiples localidades geográficas, siempre y cuando un sistema de seguimiento electrónico se utilice.
- ❖ Los Sistemas electrónicos que simulan la funcionalidad de una pared física de tarjetas están disponibles por una variedad de vendedores.
- ❖ Llevando a cabo reuniones regulares reduce el costo de la coordinación de estas reuniones y mejora la asistencia.
- ❖ La planificación de prioridades y entregas debe ser realizada de forma independiente y debe tener cadencia independiente.
- ❖ Las reuniones de pie diarias deben ser utilizadas para discutir asuntos, impedimentos y flujo. Ellos normalmente no siguen el patrón establecido por otros métodos de desarrollo Ágil.
- ❖ Las reuniones de pie diarias son una parte esencial para fomentar una cultura de mejora continua. Debido a que la reunión de pie reúne brevemente a todo el equipo cada día, dan una oportunidad para que todos los interesados propongan y discutan oportunidades de mejora. El período inmediatamente después de la reunión seguido se desarrolla en una discusión informal de mejora de procesos.
- ❖ Limpiar el *backlog* mediante un triaje regular reduce su tamaño mejorando la eficacia y la eficiencia de las reuniones de priorización.
- ❖ La gestión de asuntos, el escalamiento y la resolución es una disciplina fundamental para la mejora del rendimiento de un equipo y deben abordarse al comienzo del desarrollo del equipo.
- ❖ Las rutas de escalamiento y las políticas deben estar claramente definidas.

❖ CAPÍTULO 8 ❖

Estableciendo una cadencia de entrega

La sección 3 (capítulos 6 al 15) describe la mecánica para la implementación de un sistema kanban, concluyendo con el capítulo 15, que describe cómo empezar con una iniciativa de cambio Kanban. El permiso para empezar requiere lograr un tipo diferente de trato con el resto de las partes interesadas externas comparado con lo que es habitual entre una organización de desarrollo de software y sus socios. Parte de este nuevo tipo de negociación implica el acuerdo y el compromiso de entregas periódicas de software que funcione.

El término “cadencia de entrega” en el título de este capítulo implica establecer un patrón de entrega desoftware que funciona a un intervalo regular. Por ejemplo, si acordamos hacer una entrega cada dos semanas, tendríamos una cadencia de entrega quincenal, o 26 veces al año. Tal vez, podríamos incluso acordar en el día de la entrega. Por ejemplo, cada segundo miércoles, como fue el caso de entregas de mantenimiento de las aplicaciones de TI en Corbis.

En los círculos de desarrollo de software ágil está generalmente establecido que una cadencia regular es importante. Los métodos ágiles de desarrollo logran esto con una iteración de time-boxed, habitualmente de entre una a cuatro semanas de duración. El argumento para el time-boxed se basa en la idea de que un latido constante en un proyecto es importante. No existe una premisa fundamental que para lograr esto es necesario utilizar iteraciones estrictas de time-boxed. Al comienzo de la iteración, el alcance, o *backlog*, es acordado y el compromiso hecho.

El trabajo comienza! Una cierta cantidad de análisis, la planificación de pruebas, diseño, desarrollo, pruebas y refactorización se realiza. Si todo va bien, se realiza todo el alcance comprometido. La iteración termina con la entrega desoftware que funciona y con una reunión de retrospectiva para discutir mejoras futuras y ajustes en los procesos. Y el ciclo comienza de nuevo. Todo esto ocurre a una cadencia regular que fue acordada de antemano—semanal, quincenal, mensual u otra.

Kanban prescinde de la iteración de time-boxed y en su lugar desacopla las actividades de priorización, el desarrollo y entrega. Se permite adaptar la cadencia de cada uno a su nivel natural. Kanban no prescinde de la noción de una cadencia regular, sin embargo. Equipos de Kanban entregan software regularmente, prefiriendo una escala de tiempo corta. Kanban continúa entregando basado en los principios detrás del Manifiesto Ágil¹⁹. Sin embargo, Kanban evita cualquier disfunción introducida por forzar artificialmente las cosas en los cajones de tiempo.

Durante los últimos diez años, los equipos que usan métodos ágiles han aprendido que menos WIP es mejor que más WIP. Ellos han aprendido que las transferencias de lotes pequeños son mejores que las de lotes grandes. Como respuesta a este aprendizaje, a mediados de la década pasada adoptaron duraciones de iteración más cortas. Equipos de Scrum habituales pasaron de cuatro semanas a dos semanas y los equipos de Programación Extrema pasaron de dos semanas a una semana. Uno de los problemas que esto introdujo es que puede ser difícil analizar el trabajo en unidades lo suficientemente pequeñas para lograr que se hagan dentro de la ventana de tiempo disponible. El mercado respondió a esto mediante el desarrollo de formas más sofisticadas para analizar y escribir historias de usuario. El objetivo era reducir el tamaño de las historias para hacerlas más granulares y para reducir la variabilidad en el tamaño con el fin de encajarlos en iteraciones pequeñas. Aunque este enfoque es racional en teoría, es difícil de lograr. Y cae dentro de la categoría del sexto elemento de la Receta para el éxito: ataque la fuente de variabilidad para mejorar la previsibilidad. Como se describe en el capítulo 3, la reducción de la variabilidad a menudo requiere que la gente cambie su comportamiento y aprenda nuevas habilidades. Esto significa que es difícil de hacer.

Así que los equipos han luchado para escribir historias de usuario consistente y suficientemente pequeñas como para caber en iteraciones cortas de time-boxed. Esto ha llevado a varias disfunciones. La primera consiste en invertir la tendencia a las pequeñas iteraciones y volver a iteraciones más largas. La alternativa es escribir historias que están enfocadas en elementos de la arquitectura o alguna descomposición técnica de los requerimientos. Esto resulta en, por ejemplo, una historia para la interface de usuario, una historia para la capa de persistencia y así sucesivamente. Una segunda alternativa es romper la historia a través de tres iteraciones por etapas, en la que la primera iteración realiza análisis y tal vez la planificación de pruebas, la segunda implica el desarrollo del código y la tercera consiste en las pruebas del sistema y resolución de defectos. Cualquiera o todas estas disfunciones son

posibles. Los dos últimos hacen burla de la noción de iteraciones de time-boxed y disfrazan el hecho de que el trabajo está todavía en progreso cuando se está informando como hecho.

Kanban desacopla el tiempo que se necesita para crear una historia de usuario de la fecha de entrega. Mientras que algo de trabajo está completo y listo para su entrega, algún otro trabajo estará en progreso. Desasociando el tiempo de entrega para el desarrollo de la cadencia de entrega, tiene sentido preguntar con qué frecuencia la priorización (y tal vez la planificación y estimación) debe suceder. Parece poco probable que las discusiones de planificación, estimación y priorización deban suceder al mismo ritmo que la entrega y entrega de software. Son funciones completamente diferentes, a menudo requiriendo la atención de diferentes grupos de personas. El esfuerzo de coordinación en torno a la entrega es sin duda diferente de los esfuerzos de coordinación en torno a la priorización de trabajo nuevo. Kanban permite la disociación de estas actividades.

Kanban también desacopla la cadencia de la priorización tanto del tiempo de entrega a través del sistema así como de la cadencia de entrega. Este capítulo describe los elementos que intervienen en el acuerdo sobre una cadencia adecuada de entrega y cuándo, o si tendría sentido tener entregas bajo demanda o ad hoc en lugar de un itinerario regular de entrega. Sobre la misma línea de pensamiento, el capítulo 9 discute cómo establecer una cadencia de priorización y cuándo, o si tendría sentido tener priorización bajo demanda o ad hoc en lugar de una reunión regular. Y en el capítulo 11 se describe cómo establecer expectativas en torno a tiempo de entrega y la forma de informar sobre el contenido de una entrega.

Costos de coordinación de entrega

La coordinación de cada entrega de software tiene su costo. Es necesario reunir a la gente para discutir el despliegue (o entrega), la fabricación, el empacado, la mercadotecnia, los comunicados de mercadotecnia, la documentación, el entrenamiento para del usuario final, la capacitación de revendedores, el entrenamiento para el servicio de apoyo y servicio técnico, la documentación de instalación, los procedimientos de instalación, el personal de guardia, los horarios de base durante la entrega y así sucesivamente. La planificación de la entrega de una pieza desoftware que funcione puede ser increíblemente compleja, dependiendo de la naturaleza del dominio del negocio y el tipo de software. La actualización de un sitio web puede ser algo trivial en comparación a la actualización del firmware entregado en equipo militar distribuido por todo el mundo, en satélites en órbita, en aviones de combate, o en los nodos de una red telefónica.

En 2002, cuando estábamos planeando la entrega de la actualización del PCS Vision en la red de telefonía celular de Sprint PCS en los Estados Unidos, decenas de miles de personas tuvieron que ser entrenadas. En las tiendas de todo el país, 17.000 empleados de distribuidoras tuvieron que ser entrenados en las características de la nueva red y el

funcionamiento de alrededor de 15 nuevos modelos ofrecidos. Un número similar de personas tuvieron que ser entrenados para responder a las inevitables llamadas al soporte técnico que se producirían cuando el público tomara posesión de sus nuevos dispositivos. Tan sólo la planificación de la capacitación de alrededor de 30.000 personas tiene un costo grande en tiempo y dinero.

Por eso es importante entender los costos de coordinación asociados con hacer una entrega. Por ejemplo, si los desarrolladores de software tienen que asistir a reuniones de coordinación de entregas, ¿los distrae eso de construir el software para la entrega? A continuación sigue una lista de algunas de las preguntas a considerar.

- ¿Cuántas reuniones?
- ¿Cuántas personas?
- ¿Cuánto tiempo consumirá?
- ¿Qué costo de oportunidad se incurre cuando la gente es distraída de sus actividades regulares?

Costos de transacción de entrega

Con artículos físicos es fácil entender los costos de transacción para hacer una entrega. En primer lugar está el pago. El cliente se encargará de pagar al proveedor con algún instrumento monetario tal como una tarjeta de crédito. Por el beneficio de tomar el pago vía tarjeta de crédito, los vendedores principales tales como MasterCard y Visa le cargan al vendedor un costo de transacción, usualmente entre el dos y el cuatro por ciento del valor de la transacción.

Además de los costos de la transacción financiera entre el consumidor y el proveedor, también puede haber gastos de envío. El envío cuesta dinero, pero también cuesta tiempo y esfuerzo humano y puede haber costos de instalación. Digamos, por ejemplo, que usted compra una lavadora en Sears y arregla para que se la entreguen en un día determinado. Detrás del telón, programar la entrega y coordinar con el conductor que el modelo correcto sea entregado en la casa correcta a la hora correcta en el día correcto es un costo de coordinación de la entrega. Que el conductor recoja la máquina en el almacén, llevarla a su casa y desembalarla para usted es un costo de transacción. Tal vez la misma persona u otra persona, un plomero, lo instale para usted. El plomero necesita tiempo para ir a su casa y aún más tiempo para realizar la instalación. Todo este tiempo y esfuerzo para la entrega y la instalación es una parte del costo de transacción por comprar esa lavadora.

Económicamente, el vendedor absorbe el costo de la transacción de tarjeta de crédito. Los otros costos de transacción para la entrega e instalación a menudo se le pasan al consumidor. No todos los costos de transacción son “vistos” o “sentidos” por todos los actores de la cadena de valor afectan el rendimiento económico del sistema en su conjunto. El efecto

neto de todos estos costos es para inflar el precio final pagado por el consumidor sin tener que aumentar el valor entregado.

Es cierto que la lavadora sin la entrega o la instalación es de poco valor, pero su capacidad de valor agregado es que lava la ropa. La entrega y la instalación son actividades sin valor agregado que se deben contar como costos de transacción.

En el desarrollo de software, los costos de transacción de entrega también pueden ser de naturaleza física. Algunas empresas, como Microsoft, sigue haciendo “entrega para manufactura” (en inglés: *release to manufacture*—RTM) e imprimen medios físicos, tales como DVDs y los embalan y envían a los distribuidores, minoristas y otros socios. Con el software integrado, puede ser necesaria la fabricación de un conjunto de circuitos integrados, o, al menos, “fundir” el código de software en el firmware utilizando tecnología como EE-PROM. Los circuitos integrados, si es necesario, tendría que ser físicamente montados en el hardware que controlan.

En otros casos, puede ser posible hacer una entrega electrónica. Por ejemplo, los teléfonos celulares permiten ahora lo que se llama administración de dispositivos por aire para actualizar el firmware y la configuración del dispositivo. Muchos satélites y sondas espaciales permiten la actualización del firmware por aire. Esta capacidad de entrega suave hace las misiones espaciales mucho más ágiles de lo que eran en el pasado. La misión se puede cambiar mediante la carga de nuevo software. Los defectos también se pueden corregir *in situ*. Algunos defectos infames, como la capacidad de enfoque del telescopio Hubble, fueron corregidos (en parte) con cambios de software. Esto ha cambiado la economía de entrega.

Muchos lectores pueden estar involucrados en el desarrollo web o el desarrollo de aplicaciones internas. Las entregas pueden significar simplemente copiar archivos a través de un conjunto de discos en otras máquinas. Si bien esto puede parecer trivial, a menudo no lo es. Muchas veces es necesario planificar un procedimiento elaborado para cambiar con cuidado las bases de datos, los servidores de aplicaciones y otros sistemas, para entonces actualizarlos y reactivarlos. Uno de los mayores problemas es la migración de datos de una generación de un esquema de base de datos a otro. Las bases de datos pueden ser muy grandes. El proceso de serialización de los datos a un archivo, analizarlo (*parsing*), desembalarlo, quizás mejorarlo o aumentarlo con otros datos, volver a analizarlo y desembalarlo en un nuevo esquema puede tomar horas—incluso días.

En algunos entornos, la entrega de software puede durar horas o días. Esto no es necesariamente así porque el software sea de mala calidad o tenga una arquitectura defectuosa, sino que simplemente refleja la naturaleza del dominio en el que se utiliza el software. Todas las actividades involucradas en la distribución exitosa de software, ya sea que se trate de aplicaciones empaquetadas, firmware integrado, o aplicaciones de TI que se ejecutan en servidores internos, deben ser tomadas en cuenta, planeadas, programadas, con recursos y luego efectivamente realizadas. Todas estas actividades son los costos de transacción de hacer una entrega.

La eficiencia de la entrega

La ecuación para calcular la eficiencia de la entrega puede ser evaluada de dos maneras. La manera más simple es ver la labor y los costos involucrados. El método más complejo es considerar el valor entregado.

Primero, el modelo de sólo-costo. Debemos considerar los costos totales incurridos entre entregas. A menudo este es un monto conocido—la tasa de gasto de la organización. Si hacemos lanzamientos mensuales y nuestra tasa de gasto es \$1.3 millones por mes, nuestros costos son de por lo menos \$1.3 millones por entrega. Podemos incurrir también en costos de manufactura física, costos de imprenta, costos de publicidad y gastos de bolsillo para coordinar una entrega. Todo esto es relativamente fácil de explicar. Imaginemos que son \$ 200.000 en este caso. Así pues, nuestro costo total de entrega es de \$ 1,5 millones.

Sabemos que nuestros costos de bolsillo adicionales son de \$200,000, pero ¿que cantidad de los \$1.3 millones se gastaron en planificación, coordinación y en la ejecución actual de la entrega? Si se dispone de datos adecuados de seguimiento de tiempo podríamos ser capaces de calcularlo. Podríamos hacer una buena conjetura aún sin hacer esos cálculos. ¿Cuántas reuniones? ¿Cuántas personas? ¿Cuántas horas transcurrieron en las reuniones? Incluya el número de horas-hombre utilizadas en las actividades de entrega y en la entrega. Multiplíquelo por la tarifa por hora. Si esto se elevó a \$300,000 dólares, tendremos un costo de transacción y coordinación de \$500,000 dólares para una entrega.

$$\text{Eficiencia de Entrega\%} = 100\% \times (\text{Costo Total} - (\text{Costo de Coordinación} + \text{Costo de Transacción})) / \text{Costo Total de la entrega del Software}$$

En este ejemplo, nuestro porcentaje de eficiencia es:

$$100\% \times (\$1,500,000 - \$500,000) / \$1,500,000 = 66.7\%$$

Para ser más eficaz, tenemos que (a) aumentar el tiempo entre las entregas, o (b) reducir los costos de coordinación y transacción. La opción (a) es típica de las empresas occidentales del siglo XX. Es la elección que valora la “economía de escala”: Hacer las cosas en grandes lotes (*batches*) con el fin de amortizar los costos sobre períodos más largos de tiempo. La opción (b) es típica de los negocios japoneses de finales del siglo XX y de las empresas que persiguen el Pensamiento Lean. La opción (b) se enfoca en la reducción de desperdicio mediante la reducción de los costos de coordinación y transacción a fin de hacer eficiente el tamaño del lote; en este caso, para hacer el tiempo entre entregas eficiente.

¿Cuanta eficiencia es suficiente?

Esta es en realidad una pregunta abierta. Cada negocio tiene perspectivas distintas sobre los números adecuados de eficiencia y dependerá mucho del valor entregado.

Acordando con una cadencia de entrega

Si entendiéramos cuánto valor se tiene que entregar en una entrega podríamos hacer una mejor elección sobre la frecuencia de entrega. Si nuestra entrega mensual de software fuera para obtener \$2 millones en ingresos frente a nuestros costos de \$1.5 millones, sabríamos que estamos teniendo un margen de \$500,000 de esa actividad. Podríamos reescribir nuestra ecuación de eficiencia como:

$$\text{Eficiencia de Entrega\%} = \\ 100\% \times (1 - ((\text{Costos de Transacción} + \text{Costos de Coordinación}) / (\text{Margen} + \text{Costos de Transacción} + \text{Costos de Coordinación})))$$

En nuestro ejemplo, esto produciría un porcentaje de eficiencia de

$$100\% \times (1 - ($500,000 / ($500,000 + $500,000))) = 50\%$$

Esto se complica más porque el cálculo del valor real de una entrega es casi imposible. Puede que no tengamos órdenes fijas a precio fijo. Podemos estar especulando sobre la incorporación en el mercado así como sobre el precio y margen que podemos alcanzar. Podemos estar lanzando ítems de valor intangible, tales como una revisión de nuestro producto clave y materiales de mercadotecnia, o un conjunto de correcciones de defectos y usabilidad sobre nuestro producto o sitio web.

Calcular si debemos o no retrasarnos y lanzar con menos continuidad para ser más eficientes es igualmente difícil. Incrementar el tiempo para colocar en el mercado puede tener un efecto adverso en nuestra cuota en el mercado, nuestro precio y nuestro margen. Este concepto de eficiencia de entrega no es ciencia exacta. Lo que es más importante es que usted, el equipo y la organización estén conscientes de los costos de entrega—en tiempo y dinero—y que sean capaces de hacer algún tipo de evaluación racional sobre la frecuencia aceptable de entrega.

Si les lleva a diez personas tres días para hacer una entrega exitosa de código de un equipo de cincuenta personas, ¿es aceptable hacer una entrega cada diez días hábiles (o dos semanas naturales)? La respuesta es probablemente no. Quizá una vez al mes o veinte días hábiles es una mejor opción. Por otro lado, el mercado podría ser tal que la agilidad y tiempo de entrega al mercado es vital, donde muchos riesgos puedan mitigarse a través de entregas más frecuentes y vale la pena pagar el costo. Usted tiene que usar juicio y decidir por sí mismo.

Mejore la eficiencia para incrementar la cadencia de entrega

Para seguir el ejemplo, hemos determinado que les toma a diez personas tres días lanzar el código. De ello derivamos que las entregas mensuales son aceptables. Sin embargo, varias personas creen que con mejor calidad de código, mejor gestión de la configuración,

mejores herramientas para gestionar la migración de datos y ensayos regulares del proceso de entrega, será posible cortar el tiempo de tres días a ocho horas. Repentinamente una entrega cada dos semanas parece posible. ¿Tal vez es posible cada semana? ¿Qué hacemos?

Mi consejo es hacer una elección conservadora al principio. Acuerde entregasmensuales. Deje que la organización pruebe que puede lograr ese nivel de consistencia. Después de algunos meses, reflexione sobre la calidad del código y promueva un programa para mejorar la gestión de configuraciones. Si hay recursos disponibles, comprométalos a crear herramientas para mejorar la migración de datos a través de esquemas durante una entrega. Tal vez tenga que comprar, instalar y poner en servicio tal entorno. Todo esto tomará tiempo.

Rete al equipo y a los directores inmediatos de función, quienes controlan y efectúan entregas, a reducir los costos de transacción y coordinación. Conforme estos costos bajan, revise el progreso durante las reuniones de revisión de operaciones y establezca enlaces con otras partes interesadas. Cuando se sienta seguro de poder comprometerse a una cadencia de entrega más frecuente, tal como cada dos semanas, ¡hágalo!

La reducción de costos de coordinación y transacción está en el corazón de Lean. Es la eliminación de desperdicio en su forma más potente. Permite que los lotespequeños se hagan eficientes. Permite agilidad de negocio. La reducción de los costos de coordinación y de transacción cambia el juego. Sin embargo, no se enfoque tan solo en su reducción. Redúzcalos con una meta en mente—hacer entregas más frecuentes desoftware que funcione y por ende entregar más valor a sus clientes con más frecuencia.

Haciendo entregas bajo demanda o ad hoc

Las entregas regulares tienen sus ventajas. Hacer la promesa de entregar en fechas específicas, por ejemplo, cada segundo miércoles, le permite a aquellos involucrados programar alrededor de ello. Provee certidumbre. También puede reducir los costos de coordinación porque no hay sobrecarga debido a la decisión de cuando hacer la entrega y quién tiene que participar—todo eso se establece una vez y es bastante consistente en adelante.

Las entregas regulares también ayudan a generar confianza. La carencia de previsibilidad destruye la confianza. La falla al hacer una entrega en una fecha prometida se toma en cuenta mucho más que el contenido específico de la entrega misma.

Habiendo fortalecido la cadencia regular de entregas, ¿tiene sentido tener entregas bajo demanda o ad hoc bajo algunas circunstancias? ¿Cuáles podrían ser esas circunstancias?

Primeramente, las entregas bajo demanda o ad hoc tiene sentido cuando los costos de coordinación de la actividad de entrega son bajos. Cuando los costos de coordinación son bajos no hay beneficio en programar actividades de coordinación regularmente. En segundo lugar, tiene sentido cuando los costos de transacción son bajos, tal vez porque la entrega del código está grandemente automatizada y la calidad está asegurada de antemano, antes de la entrega. Finalmente, tiene sentido en ambientes donde las entregas

son tan frecuentes que no hay necesidad real de desarrollar un patrón: el software nuevo se entrega tan seguido como para parecer continuo para la mayoría de los observadores y partes interesadas externas—sus cerebros no han sido programados para anticipar una fecha de entrega. Y cuando no hay expectativa no puede haber decepción.

Este tipo de entrega quasi-continua de código parece ser útil y necesaria en algunas industrias. Los ejemplos que han surgido de quienes adoptaron Kanban inicialmente han sido primordialmente en la industria de los medios de comunicación, por ejemplo, IPC Media en Londres, donde utilizan múltiples sistemas kanban para planificarel desarrollo de propiedades de medios en línea tales como mousebreaker.com, un juego en línea muy adictivo.

Las primeras dos circunstancias de baja coordinación y bajos costos de transacción tienden a indicar organizaciones de alta madurez. Esto ha sido observado también con los adoptantes iniciales de Kanban. El departamento de XIT de Microsoft estaba asociado con un vendedor con CMMI-L5 en la India y con Microsoft TI en Redmond, Washington, el cual está aproximadamente en CMMI-L3 de madurez. Organizaciones muy maduras tienden a establecer niveles de confianza con socios de cadena-de-valor y partes interesadas internas, incluyendo la alta dirección, por lo que no tienen una cadencia regular de entrega para generar esa confianza.

Por lo que, en general, elija una cadencia de entrega regular excepto bajo circunstancias en las que la confianza ya existe, altos niveles de capacidad y madurez ya existen y en dominios donde la entrega quasi-continua es deseable.

Hay una última circunstancia en la cual es aceptable hacer entrega bajo demanda. Cuando hay una solicitud urgente que está siendo tratada como un caso especial y se acelera. Este concepto de la clase de servicio *Expreso* está explicado en la sección de Tiempo de Entrega del capítulo 11. Podemos elegir Expreso por varias razones, la primera y más obvia es la existencia de un defecto crítico en producción. En circunstancias donde nada importa más que resolver el problema, debe plantearse una entrega fuera de ciclo.

Hay otras circunstancias bajo las cuales una entrega fuera de ciclo pude tener sentido. Probablemente el equipo de ventas recibió un pedido de un cliente grande que desea una versión personalizada del software y necesitan la entrega antes de fines de mes (y del trimestre) debido a restricciones de presupuesto y del ciclo fiscal. El pedido llega del grupo de operaciones que el grupo de ingeniería de software tiene que poner a un lado todo lo que tienen para satisfacer el pedido de este cliente porque tiene un gran valor en cuanto a los ingresos.

Bajo estas circunstancias tiene sentido planificaruna entrega especial fuera de ciclo. Esta entrega debe ser tratada como excepcional y la cadencia de entrega regular debe ser re-establecida lo antes posible después de la entrega excepcional. Sin embargo tiene beneficio usar el sentido común. Por ejemplo, si una entrega regular está programada para el miércoles y la entrega excepcional se requiere el viernes de la misma semana, puede tener

sentido retrasar la entrega del miércoles hasta el viernes. Si usted elige hacer esto, es importante comunicarlo adecuadamente y lo suficientemente pronto para que se ajusten las expectativas. Usted no desea perder la confianza de sus socios de la cadena-de-valor como efecto colateral por intentar ser servicial y útil.

Para llevar

- ❖ “Cadencia de Entrega” significa un intervalo regular acordado entre entregas desoftware que funcione.
- ❖ Kanban desacopla la cadencia de entrega tanto del tiempo de entrega del desarrollo como de la cadencia de priorización.
- ❖ Iteraciones cortas, de time-boxed, han conllevado a disfunción con equipos que intentan adoptar métodos de desarrollo ágil.
- ❖ La entrega o entrega de software involucra la coordinación de mucha gente con diversas funciones. Toda esa coordinación tiene un costo medible.
- ❖ La entrega o entrega de software conlleva un conjunto de costos de transacción en tiempo y dinero. Estos costos pueden ser determinados y rastreados.
- ❖ La eficiencia de entrega puede ser calculada comparando la suma de los costos de coordinación y transacción para hacer una entrega contra el costo total de la creación del software para esa entrega.
- ❖ La cadencia de Entrega puede establecerse comparando los costos de producción de la entrega contra el valor recibido por esa entrega.
- ❖ La eficiencia y la cadencia pueden incrementarse enfocándose en la reducción de los costos de transacción y coordinación.
- ❖ Las entregas regulares generan confianza.
- ❖ El establecimiento de la expectativa de entregas regulares, y entregando consistentemente conforme a la expectativa, puede ser adictivo.
- ❖ Programando entregas regulares se reducen los costos de coordinación.
- ❖ Las entregas ad hoc o bajo demanda pueden tener sentido para organizaciones altamente maduras con altos niveles de confianza y bajos costos de transacción y coordinación establecidos.
- ❖ Las solicitudes legítimas de entregas aceleradas también pueden ser causa de una entrega fuera de ciclo. Entregas regulares deben restablecerse lo antes posible después de tal caso especial y excepcional.

❖ CAPÍTULO 9 ❖

Estableciendo una cadencia de entrada

Este capítulo discute los elementos que intervienen en el acuerdo sobre una cadencia adecuada de priorización y cuándo, o si, tendría sentido tener priorización bajo demanda o ad hoc en lugar de una reunión de priorización programada regularmente.

Costos de coordinación de priorización

Cuando estábamos introduciendo Kanban en Corbis, en 2006, escogimos comenzar con el esfuerzo de ingeniería de mantenimiento. Su función era gestionar las solicitudes de actualizaciones menores y resolución de defectos en producción para toda la gama de aplicaciones de TI, incluyendo funciones tales como finanzas y recursos humanos, así como los sistemas más específicos al negocio del sistema de manejo de gestión de activos digital y el sitio web para comercio electrónico. Estos sistemas servían a por lo menos seis unidades de negocio, incluyendo ventas, mercadotecnia, operaciones de venta, finanzas y la función que apoya la cadena de suministro que vendía fotografía digital, etiquetado de metadatos, catalogación y cumplimientos—en esencia, la cadena de suministro del negocio.

Seis departamentos competían por los recursos disponibles para hacer estos cambios y actualizaciones pequeñas. Cuando el sistema kanban fue introducido inicialmente, se hizo un caso de negocio se hizo para una función de ingeniería de mantenimiento que pudiera brindar entregas

tácticas frecuentes. Esta función de mantenimiento le permitiría crear una agilidad limitada de negocio a través de la entrega de características menores incrementales mientras que otras aplicaciones de TI nuevas eran construidas utilizando un mecanismo de gobierno a través de una oficina de gestión de programas (en inglés: *Program Management Organization—PMO*). Cada proyecto en la cartera estaba justificado y autorizado basado en su propio caso de negocio. La función de mantenimiento había sido aprobada por el comité ejecutivo y un fondo adicional del 10% autorizado por la función de ingeniería de software, que se tradujo en un incremento de personal de 5 personas para el departamento. A esta nueva capacidad se le llamó el Equipo de Respuesta Rápida (en inglés: *Rapid Response Team—RRT*). El nombre era poco apropiado pues inicialmente no era ni rápido, ni respondía y no se tenía un equipo.

Crear un departamento especial para mantenimiento no era factible con esas cinco personas. Corbis tenía una diversidad considerable de sistemas de TI y muchos requerían habilidades especializadas. La función de análisis que desarrollaba y elaboraba requerimientos de sistema se sustentaba fuertemente en los especialistas. Las cinco personas adicionales se distribuyeron más o menos uniformemente a través de la función de ingeniería de software que incluía la gestión de proyectos, el análisis de sistemas, el desarrollo, las pruebas, la administración de configuración, y la ingeniería de *builds*. Por lo que no se tenía un equipo per se. La T en RTT no tenía significado. El reto para la dirección era mostrar que este diez por ciento adicional para recursos era utilizado en trabajo de mantenimiento y soporte y no estaba siendo absorbido en trabajo de la cartera de proyectos principales.

Se decidió entonces dedicar un director de proyecto al esfuerzo de ingeniería de mantenimiento. Mientras que ella no estaba dedicada de tiempo completo a la ingeniería de mantenimiento, brindó un punto focal para la comunicación y coordinación y contó como la mitad de una de las cinco personas asignadas a esa iniciativa. Un ingeniero de *builds* del equipo de gestión de configuración también fue asignado específicamente a esa iniciativa. Sus labores eran mantener los sistemas de pre-producción requeridos para pruebas y staging, y para construir código y empujarlo en el entorno de pruebas cuando fuese requerido.

A fin de mantener la integridad del entorno compartido de pruebas, el cual era usado por múltiples proyectos al mismo tiempo, Corbis tenía la política de que solamente se les permitía a los ingenieros de *builds* promover código del entorno de desarrollo al entorno de pruebas. Esto cambiaría más tarde, pero en septiembre de 2006 la realidad era que un ingeniero de *build* era requerido para promover código a pruebas.

Antes de introducir Kanban el esfuerzo de coordinación requerido para acordar en el alcance de una entrega de mantenimiento era prohibitivo. El director de proyecto y a menudo su jefe, el director de proyecto del grupo, convocaban una reunión con todas las partes relevantes, incluyendo al analista de negocio, representantes del negocio, analistas de sistemas, directores de desarrollo, líderes de pruebas y el ingeniero de *builds* y a veces el director de administración de configuración así como personal de operaciones

de sistemas y de asistencia. Estas reuniones podían tomar varias horas y a veces no eran concluyentes. Los miembros del equipo eran enviados a hacer estimaciones y se programaba otra reunión. La reunión de seguimiento continuamente se convertía en debates sobre prioridades y, de nuevo, normalmente no era concluyente. En septiembre de 2006 estaba tomando dos semanas hábiles, con varias reuniones de muchas horas llegar a un acuerdo en el alcance para una entrega que se suponía tomaría tan solo dos semanas en construir y entregar. Debido a que la longitud de la iteración era de dos semanas, solamente solicitudes muy pequeñas podían ser acomodadas y muchas solicitudes potencialmente valiosas eran ignoradas. Dichas solicitudes eran redirigidas a un proyecto grande por lo que podrían tomar meses o años antes de ser implementadas. El sistema no era ni rápido ni respondía antes de la introducción de un sistema kanban. Por lo que las RR de RRT tampoco tenían sentido.

Kanban liberó a este equipo de todas estas disfunciones y la iniciativa se ganó con todo derecho el nombre de Equipo de Respuesta Rápida (RRT).

Al introducir el sistema kanban, los propietarios de negocio fueron educados en cuanto al flujo, la cola de entrada y el mecanismo de arrastre. Ellos aprendieron que se les pedía que aprovisionaran los espacios vacíos en la cola y que no fuese necesario priorizar el *backlog* de solicitudes. Si había dos espacios libres en la cola, la pregunta sería, “¿Qué dos nuevos ítems quiere a continuación?” Asumiendo que contamos con datos sobre el rendimiento del tiempo de entrega promedio y la fecha límite contra el tiempo de entrega especificado en el acuerdo de nivel de servicio, la pregunta entonces será más elaborada, tal como, “¿Qué dos ítems te gustaría más que te entregáramos en 30 días a partir de ahora?” El reto era entonces para seis propietarios de negocio el competir de alguna manera para acordar y escoger dos ítems de las múltiples elecciones posibles.

Sin embargo, la pregunta es simple y se sugirió que contestar tal pregunta simple debería lograrse dentro de una hora. Había consenso en que una hora era razonable, por lo que se les preguntaba a los propietarios de negocio si estarían dispuestos a asignar una hora de su semana para atender una reunión de priorización para aprovisionar la cola de entrada para la función de ingeniería de mantenimiento.

Acordando en una cadencia de priorización

La reunión se programó para todos los lunes a las 10 AM. Era atendida generalmente por directores de alto rango de los que solían atender las reuniones de alcance de entrega, usualmente el vicepresidente de cada grupo funcional. Adicionalmente atendían el director de proyecto, el Director Senior de Ingeniería de Software, el Director Senior de Servicios de TI (cuyas responsabilidades incluían gestión de proyectos), por lo menos un director de desarrollo, el director de pruebas, el director del equipo de análisis y ocasionalmente algunos contribuyentes individuales.

Acordar en una cadencia regular les provee a todos de previsibilidad. Fueron capaces de reservar una hora los lunes por la mañana y la reunión era bien atendida en general.

La cadencia semanal es buena para la priorización. Provee una interacción frecuente con los propietarios del negocio. Genera confianza a través de la interacción involucrada. En el juego cooperativo colaborativo del desarrollo de software les permite a los jugadores mover las piezas una vez a la semana. Las reuniones semanales son posibles debido a la simplicidad de la pregunta que debe ser contestada y a la garantía de que la reunión puede ser completada en una hora. Cuando la gente de negocio toma tiempo fuera de sus negocios necesitan ver que su tiempo está bien utilizado.

Muchos de los aspectos de Kanban contribuyen a hacer que la reunión semanal de priorización sea gratificante: es una experiencia colaborativa; hay transparencia en el trabajo y en el flujo de trabajo; el progreso puede ser reportado cada semana; y todos sienten que están contribuyendo a algo valioso. Varios de los vicepresidentes en Corbis sentían que el proceso de RRT estaba haciendo diferencia. Obtuvieron un nuevo nivel de respeto hacia el departamento de TI y aprendieron a colaborar con sus pares en otros departamentos de una manera que nunca antes había sido común en Corbis.

Eficiencia de la priorización

Puede que las reuniones semanales de coordinación no sean la respuesta correcta para su organización. Usted se dará cuenta que sus retos de coordinación son o más difíciles o más sencillos que los de Corbis. Algunos equipos se sientan juntos, por lo que no hay necesidad de reuniones; la coordinación de prioridades se suma a una discusión rápida al otro lado del escritorio. Por otro lado, algunos equipos pueden tener gente en zonas horarias distintas y en diferentes continentes, por lo que las reuniones semanales puede que no sean programadas tan fácilmente. Probablemente la pregunta a contestar no será tan simple como en el ejemplo de Corbis y la reunión tomará más tiempo. Es difícil imaginar una situación con más de seis grupos compitiendo por el mismo recurso compartido, pero es posible. Entre más grupos están involucrados, puede que las reuniones sean más largas. Y entre más larga sea la reunión, es más probable que sea menos frecuente.

Como consejo general, priorización más frecuente es deseable. Ello permite que la cola de entrada sea más pequeña y, como resultado, hay menos desperdicio en el sistema. El WIP es más bajo y por lo tanto el tiempo de entrega es más corto. Priorización más frecuente le permite a todas las partes trabajar juntas más seguido. La experiencia del trabajo colaborativo genera confianza y mejora la cultura. Esfuércese por encontrar el esquema de coordinación más pequeño y eficiente posible y lleve a cabo reuniones de priorización tan seguido como sea razonable.

Costos de transacción de la priorización

Con el fin de moderar una reunión eficiente todos los lunes, Diana Kolomiyets, la directora de proyecto, generalmente enviaba un email el jueves o viernes previo para informar a los participantes del número estimado de espacios vacíos anticipados a estar en cola para el lunes en la mañana. Les pedía que hojearan el *backlog* de solicitudes y seleccionaran candidatos probables para la selección del lunes. Esta “tarea” a menudo los llevó a preparar algún argumento para soportar la elección exitosa de su ítem favorito. Comenzamos a ver documentos de soporte en las reuniones de los lunes. Algunas personas preparaban un caso de negocio, algunos una presentación para argumentar su elección. Otros comenzaron a moverse entre ellos. Era probable que alguien en el consejo de priorización llevara a alguien a comer el viernes específicamente para obtener apoyo a su elección durante la reunión del lunes. Se introdujo un tira y afloja de manera tal que un miembro del consejo estaría de acuerdo en apoyar el ítem de otro esa semana a cambio del apoyo del otro miembro en una semana futura. La naturaleza de las reglas del juego, donde múltiples organizaciones compiten por el recurso compartido del RRT introdujo un nuevo nivel de colaboración.

Ocasionalmente, al negocio le preocupaba que una solicitud fuera muy costosa de implementar en comparación con su valor, por lo que podrían solicitarle al equipo de análisis que hiciera una estimación. Más adelante se introdujeron las reglas concernientes a clase de servicio para guiar si valía la pena estimar o no un ítem. Esto se explica plenamente en el capítulo 11.

Todas estas actividades, incluyendo la estimación, la preparación del plan de negocio y la selección de candidatos a partir del *backlog*, son trabajos de preparación para la priorización. Es deseable mantener bajos estos costos. Si los costos de transacción se hacen onerosos, sin importar que tan bajos sean los costos de coordinación, entonces el equipo no deseará reunirse regularmente. Evitando estimaciones detalladas lo más posible los costos de transacción se minimizan, lo que facilita reuniones de priorización más frecuentes.

Mejore la eficiencia para incrementar la cadencia de la priorización

En general, el equipo de gestión debe estar consciente de todos los gastos de coordinación y transacción incurridos por todos—no solo el equipo de desarrollo—los involucrados en priorización y selección de nuevos ítems a poner en cola para desarrollo y entrega.

Muchas organizaciones ágiles utilizan una forma de priorización llamada Póker de Planeificación, la cual utiliza una técnica de “sabiduría colectiva” en la que todo miembro del equipo vota usando una carta que representa un número de tamaño. Los votos son promediados o a veces se busca un consenso mediante la discusión de los extremistas en

los votos para entonces votar de nuevo hasta que todos en el equipo estén de acuerdo con una estimación. Las barajas de póker a menudo utilizan una escala numérica no-lineal, tal como la serie de Fibonacci, para fomentar la idea de tamaño relativo.

Algunos argumentarán que ésta técnica de planificación, la cual es también una forma de juego cooperativo colaborativo, es muy eficiente ya que permite establecer rápidamente una estimación bastante exacta. Existe evidencia anecdótica para apoyar esto, pero también hay pruebas para sugerir que el pensar-del-grupo también es posible. He oído informes de equipos tales como una empresa nueva en San Francisco donde consistentemente subestimaban a pesar de usar un juego de colaboración transparente, como el Póker de Planificación. También he escuchado de los altos directivos de un sitio web de reserva de viajes muy conocido donde los equipos siempre sobreestimaban, a pesar de usar el Póker de Planificación. Independientemente de si usted cree que estos juegos de planificación son eficaces o no, el argumento de que son eficientes vale la pena considerarlos con mayor profundidad.

Es cierto que los juegos de planificación involucran a un equipo entero para hacer la estimación de un ítem individual, tal como una historia de usuario, muy rápida. Sin embargo, el ejercicio involucra al equipo completo. Esto tiene un costo significativo de coordinación. Funciona efectivamente en grupos pequeños enfocados en un solo producto; sin embargo, si extrapolamos la técnica a una organización como Corbis, donde estamos haciendo mantenimiento sobre 27 sistemas de TI utilizando 55 personas, muchos de ellos especialistas en un área, dominio, sistema, o tecnología, sería necesario tener casi a todas las 55 personas en la reunión para hacer una buena estimación y alcanzar la “sabiduría colectiva”. Los costos de transacción de la planificación y estimación pueden ser pequeños, pero los costos de coordinación son altos.

En general, estos métodos de planificación ágil sólo son eficientes para equipos pequeños enfocados en sistemas y líneas de producto simples debido a su impacto en el costo de coordinación.

Escogiendo eliminar la estimación para la mayoría de las clases de servicio, tanto los costos de transacción como los costos de coordinación de la priorización son reducidos. Esta reducción hace más fácil llevar a cabo reuniones de priorización más frecuentes porque las reuniones permanecen eficientes. Esto le ha permitido a equipos de Kanban hacer priorización ad hoc o bajo demanda.

Efectuando priorización bajo demanda o ad hoc

Como se describe en el capítulo 4, en 2004 Dragos Dimitriu introdujo un sistema kanban en su equipo de Ingeniería de mantenimiento de XIT en Microsoft. Los socios en niveles anteriores eran cuatro directores de producto que representaban varias unidades de

negocio. Ellos enfocaban y priorizaban solicitudes de cambio para alrededor de 80 sistemas de TI soportados por el XIT.

Cuando Dragos y yo diseñamos el sistema kanban a introducir en el XIT, diseñamos una cola de entrada lo suficientemente grande para hacer frente por al menos a una semana de rendimiento. A pesar del hecho de que todos los representantes de negocio y Dragos estaban basados en Redmond, Washington, en el campus de Microsoft, la reunión de planificación se llevaba a cabo por teléfono. El campus de Microsoft es enorme. Los números de los edificios llegan a cientos, aunque de hecho hay solamente alrededor de 40 edificios en total. El área cubierta es de varios kilómetros cuadrados y el transporte entre secciones del campus se hace con minibuses o Toyota Prius. Muchos “softies” prefieren llamadas de conferencia, en lugar de cara-a-cara, para las reuniones de coordinación. Esto tiene un impacto negativo en el nivel de confianza y capital social de la fuerza de trabajo, pero incrementa la eficiencia.

Debido a ello, Dragos estableció una llamada semanal para priorizar nuevas solicitudes de cambio en el *backlog*. Los cuatro directores de producto representaban unidades de negocio que proveían fondos mediante transferencia presupuestaria entre empresas para financiar al equipo de Dragos. Basado en ese financiamiento era posible determinar más o menos cuantas veces alguien podría seleccionar un ítem del *backlog*. El director de producto que proporcionaba el 60% del soporte financiero seleccionaba en tres de cada cinco oportunidades. Los otros llegaban a seleccionar ítems de una manera similar basados en su nivel de financiamiento. El director de producto que proveía menos financiamiento llegaba a elegir 1 de cada 11 veces. Describimos esto como un método de selección de round-robin ponderado.

Por ende, las reglas del juego de colaboración cooperativa de priorización del XIT eran simples. Cada semana los directores de producto llenaban los espacios abiertos en la cola de entrada—normalmente tres espacios. Cada uno de ellos llegaría a escoger en base a su posición en la cola del round-robin. El tiempo de entrega promedio en el acuerdo del nivel de servicio era de 25 días. Así que si tenían una oportunidad de escoger una solicitud de cambio para desarrollo, se preguntarían a sí mismos, “¿Cuáles de los ítems en mi *backlog* deseo más que sea entregado en 25 días a partir de ahora?” El orden en que llegaban a escoger era muy claro y simple basado en su nivel de financiamiento para el departamento.

Debido a la naturaleza simple de estas reglas, la reunión terminaba muy rápidamente. Se hizo claro que una llamada telefónica de coordinación no era realmente necesaria. Dragos hizo que la base de datos de Microsoft Product Studio (un precursor de Visual Studio Team System, Team Foundation Server) le enviara un correo electrónico a partir de un disparador que le indicara cuando un espacio era entregado. Él entonces reenviaba el correo a los cuatro directores de producto. Ellos rápidamente acordaban quién tenía el siguiente turno para elegir un ítem y esa persona seleccionaba algo. Normalmente, un espacio vacío en la cola tenía algo asignado dentro de las siguientes dos horas.

Los costos de coordinación excepcionalmente bajos, acoplados con los costos de transacción bajos relacionados con la decisión de no estimar las solicitudes de cambio y junto a una madurez relativamente alta del equipo involucrado, le permitieron al XIT de Microsoft deshacerse de las reuniones de priorización programadas regularmente.

Vale la pena notar que Microsoft en Redmond está en el equivalente a una organización CMMI-de nivel 3 y que el equipo del vendedor utilizado para desarrollo y pruebas de XIT era CMMI-de nivel 5 situado en Hyderabad, India. Por lo que este equipo tenía las ventajas de bajos costos de coordinación, bajos costos de transacción y niveles particularmente altos de madurez organizacional. El efecto neto de los tres significó que las reuniones de priorización bajo demanda hicieron al equipo más efectivo.

Como regla general, usted debe escoger entre priorización bajo demanda o ad hoc cuando tenga un nivel de madurez organizacional relativamente alto, bajos costos de transacción y bajos costos de coordinación de priorización. De lo contrario, es mejor usar una reunión de priorización programada regularmente y coordinar la selección de los ítems para la cola de entrada con una cadencia regular.

Para llevar

- ❖ “Cadencia de priorización” significa un intervalo regular acordado entre reuniones para priorizar nuevo trabajo en la cola de entrada para desarrollo.
- ❖ Kanban elimina la disfunción potencial alrededor de la coordinación de la planificación de iteración en métodos ágiles desligando la cadencia de priorización del tiempo de entrega y de la entrega de desarrollo.
- ❖ La priorización de solicitudes de nuevo trabajo, tal como historias de usuario, involucra la coordinación de mucha gente de varias funciones. Toda esta coordinación tiene un costo medible.
- ❖ La estimación para facilitar decisiones de priorización representa los costos de transacción tanto en tiempo como en dinero asociados con la priorización. Estos costos pueden ser determinados y rastreados.
- ❖ Las políticas concernientes al método de priorización y las entradas para la toma de decisiones representan las reglas del juego de colaboración cooperativa para priorizar Kanban aplicado al desarrollo de software.
- ❖ Los juegos de planificación usados en métodos ágiles no escalan fácilmente y pueden representar un costo de coordinación significativo para equipos grandes sin enfoque más amplio que una sola línea de producto.
- ❖ La cadencia de priorización puede establecerse motivando a aquellos involucrados en la toma de decisiones de priorización a reunirse tan regularmente como sea razonable basándose en los costos de transacción y coordinación involucrados.
- ❖ La eficiencia y cadencia de priorización puede incrementarse enfocándose en reducir sus costos de transacción y coordinación.
- ❖ Las reuniones frecuentes de priorización generan confianza.
- ❖ Programar reuniones frecuentes de priorización reduce los costos de coordinación y es particularmente útil en organizaciones con baja madurez.
- ❖ La priorización ad hoc o bajo demanda puede tener sentido para organizaciones de alta madurez con altos niveles de confianza establecidos y con costos bajos de transacción y coordinación asociados con las políticas para la toma de decisiones de priorización.

❖ CAPÍTULO 10 ❖

Estableciendo los límites del Trabajo-en-Progreso

Como se discutió en el capítulo 2, una de las cinco propiedades fundamentales del Método Kanban es que el trabajo en progreso está limitado. Por ende, es correcto decir que una de las decisiones más importantes que usted tomará al introducir Kanban es escoger los límites del trabajo-en-progreso a través del flujo de trabajo.

El capítulo 15 aconseja que los límites del trabajo-en-progreso sean acordados por consenso con las partes interesadas de los niveles anteriores y posteriores de la organización y con la alta dirección. Es cierto que los límites pueden ser declarados unilateralmente; sin embargo, da poder ganar consenso y obtener un compromiso de parte de las partes interesadas exteriores con respecto a la política del límite del

tiene poder. Cuando su equipo y el proceso estén bajo estrés usted puede regresar al convenio colaborativo alrededor del cual hay consenso. Usted puede redirigir la discusión hacia una redefinición del proceso en lugar de acordar el doblar, estirar, o de alguna otra forma usar mal el sistema que se tiene diseñado e implementado. Construir un consenso es una manera de mantener la disciplina del límite del WIP y evitar tener que anular o abandonar el límite.

Límites para las tareas de trabajo

Con el equipo de XIT en Microsoft, Dragos Dimitriu decidió que los desarrolladores y ingenieros de pruebas deberían trabajar en un solo

ítem a la vez. No habría multitareas. Esto fue declarado unilateralmente, pero afortunadamente esta elección no fue problemática para ninguna de las partes interesadas. Esto se alineaba con prácticas de trabajo actuales y el método de Proceso de Software Personal (PSP) que el equipo ya utilizaba. La organización era lo suficientemente madura para mantener disciplina y seguir el proceso que se había acordado. Tal vez recuerde que al inicio del caso de estudio, en el otoño de 2004, había tres desarrolladores y tres ingenieros de pruebas en el equipo. Por ende, el límite de WIP para cada uno de ellos, desarrollo y pruebas, era de tres.

En Corbis en 2006 tomamos una decisión similar con la iniciativa de ingeniería sustentable; que los analistas, desarrolladores y ingenieros de pruebas trabajarían en tan solo un ítem de trabajo a la vez valorado por el cliente. Con nuevos proyectos grandes tendíamos a tomar decisiones diferentes. Había más trabajo colaborativo en esos proyectos. Era común que equipos de dos o tres personas trabajaran reuniones en un mismo ítem. Debido a que esos ítems podían llegar a bloquearse o retrasarse, especulamos que tendría sentido permitir algún grado de cambio de tareas y algún paralelismo adicional; por lo que el límite de WIP fue establecido para inferir dos o tres personas por ítem y permitir algún sobreflujo. Por ejemplo, si teníamos diez personas y anticipamos dos personas por ítem, el límite de WIP podría ser cinco más algunos más para mitigar el impacto de un bloqueo. Tal vez ocho (cinco más tres) sería el límite correcto para tales circunstancias.

Se ha hecho alguna investigación y observación empírica para sugerir que dos ítems en progreso por trabajador de conocimiento son óptimos. Este resultado es frecuentemente referido para justificar multitareas. Sin embargo, yo creo que esta investigación tiende a reflejar la realidad de trabajo en las organizaciones observadas. Hay muchos impedimentos y razones para que el trabajo se retrase. La investigación no reporta la madurez organizacional de las empresas estudiadas y tampoco correlaciona los datos con ninguna de los factores externos occurrentes (variaciones de causa assignable, discutidas en el capítulo 19). Por lo tanto, el resultado puede ser una consecuencia de los ambientes estudiados y no un número ideal en realidad. Sin embargo, usted puede encontrar resistencia a la noción de que un ítem por persona, par, o equipo pequeño es la elección adecuada. Se puede argumentar que tal política es demasiado restrictiva. En ese caso, establecer un límite de WIP de dos ítems por persona, par, o equipo es razonable. Puede que hasta existan casos donde un límite de tres por persona, par, o equipo es aceptable.

No hay una fórmula mágica para la elección. Es importante recordar que el número puede ser ajustado empíricamente. Usted puede seleccionar un número y observar si funciona bien o no. Si no, ajústelo hacia arriba o hacia abajo.

Límites para las Colas

Cuando trabajo es completado y está esperando a ser tomado por el siguiente estado en su flujo de trabajo, se dice que esta “en cola”. ¿Que tan grandes deben ser estas colas? Lo más pequeño posible. El límite de WIP para una cola a menudo se agrupa con su paso precedente de trabajo. Por ejemplo, las colas de Desarrollo (*Development*) y de Desarrollo Completado (*Development Done*) están agrupadas, como se muestra en la figura 10.1. Si una política estricta sobre el WIP del trabajo de tareas fue establecida, tal como estrictamente un ítem por persona, par, o equipo pequeño, será necesario tener alguna cola para absorber la variación y mantener el flujo. Si, una vez en operación, su sistema kanban sufre de comportamiento de paro-e-inicio que hace que los trabajadores estén ociosos debido a la variabilidad en la longitud de tiempo que toma completar las tareas, puede que necesite incrementar el tamaño de las colas. Sin embargo, si usted ya hizo la elección de tener, por ejemplo, dos elementos en progreso por persona, par, o equipo pequeño, usted ya tiene buffer para la variabilidad, por lo que el tamaño de la cola puede ser efectivamente cero. Simplemente agrupe la columna del trabajo de tareas de la cola de completado reuniones.

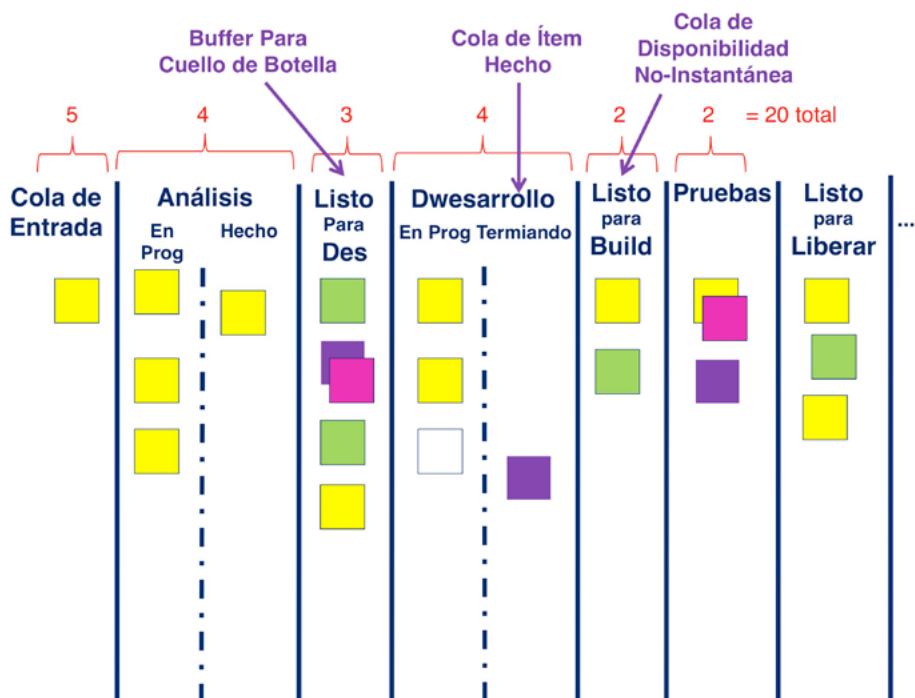


Figura 10.1 Pared de tarjetas mostrando diferentes tipos de colas y un buffer

Buffers de cuellos de botella

El cuello de botella en su flujo de trabajo puede requerir un buffer frente a él, como se muestra en la Figura 10.1. Este es un mecanismo habitual de explotación de cuello de botella. Nuevamente, usted desea que sea lo más pequeño posible. Los buffers y colas agregan trabajo-en-progreso a su sistema y su efecto el alargamiento del tiempo de entrega. Sin embargo, los buffers y colas suavizan el flujo y aumentan el rendimiento, por lo que más trabajo es entregado mediante un sistema kanban. Los buffers también aseguran que la gente continúa trabajando y proporcionan mayor utilización. Necesita haber equilibrio y los buffers ayudan a mantenerlo. En muchas instancias usted busca agilidad de negocio mediante tiempos de entrega más cortos y mayor calidad parcialmente por medio del menor monto de trabajo-en-progreso. Sin embargo, no sacrifique la previsibilidad con el fin de lograr agilidad o calidad. Si el tamaño de las colas y los buffers son demasiado pequeño y su sistema sufre de mucho comportamiento de paro-e-inicio debido a la variabilidad, los tiempos de entrega serán impredecibles, con un amplio grado de variabilidad. La clave para escoger un límite de WIP para un buffer es que debe ser lo suficientemente grande para asegurar un flujo suave en el sistema y evitar tiempo de ocio en el cuello de botella. Más detalle sobre el tamaño del buffer y como diseñar buffers tanto para cuellos de botella de capacidad restringida y disponibilidad no instantánea se discute en el capítulo 16.

Tamaño de la cola de entrada

El tamaño de la cola de entrada puede ser determinado directamente de la cadencia de priorización, o tasa de producción, en el sistema. Por ejemplo, si un equipo está produciendo a una tasa media de cinco ítems de trabajo hecho por semana (con un rango habitual de cuatro a siete ítems por semana) y la cadencia de aprovisionamiento de la cola es semanal, el tamaño de la cola debería ser de siete. De nuevo, esto puede ser ajustado empíricamente. Si corre su sistema por varios meses y la cola nunca está completamente agotada antes de la siguiente reunión de priorización, probablemente es muy larga por lo que debe reducirla por uno y observar los resultados. Repita este proceso hasta que tenga una reunión de priorización en la que les está preguntando a los representantes de negocio que surtan la cola completa.

Si, por otro lado, usted tiene una reunión semanal de priorización en un lunes y la cola se agotó el jueves por la tarde—y algunos de los miembros del equipo estuvieron ociosos como resultado de ello—la cola es muy pequeña. Incremente el tamaño de la cola por uno y obsérvelo por varias semanas.

El tamaño de la cola y el buffer deberían ser ajustados empíricamente como sea requerido. Por ello, no se preocupe por la decisión de establecer un límite de WIP. No se demore en desplegar de su sistema kanban porque no puede concordar en los números perfectos

para limitar el WIP. ¡Elija algo! Elija progresar con información imperfecta para entonces observar y ajustar. Kanban es un proceso empírico.

¿De qué tamaño debe ser la cola de entrada si usted está utilizando priorización bajo demanda? Usted recordará que el equipo de XIT en el capítulo 4 tenía una cola de entrada de cinco ítems. Esto estaba diseñado para ser lo suficientemente grande para absorber el rendimiento de una semana. Estaba basado en la asunción de que la reunión de priorización sucedía semanalmente. Sin embargo, los directores de producto decidieron rápidamente que la reunión era innecesaria y que hacer decisiones basadas en eventos cuando un espacio en la cola se hacia disponible era aceptable. Una vez que esto sucedió debí haber aconsejado a Dragos que redujera la entrada de cinco a uno. El no haberlo hecho es un reflejo de mi falta de experiencia en ese entonces. El sistema había cambiado. Las asunciones bajo las cuales había sido diseñado habían cambiado. La política sobre el tamaño de la cola de entrada estaba basada en esas asunciones y debió haber sido revisada. Si hubiésemos hecho eso, la mejora de los tiempos de entrega habría sido aún más impresionante.

Cuando el XIT cambió a priorización bajo demanda normalmente les lleva dos horas para aprovisionar un espacio vacío en la cola. Habría sido aceptable asumir que el tiempo más largo para aprovisionar la cola habría sido de cuatro horas. Sin embargo, los desarrolladores no estaban co-localizados con los directores de producto. Los tomadores de decisión sobre la priorización estaban en Redmond, Washington, y los desarrolladores estaban en Hyderabad, India. Cada uno de ellos estaba en turno trabajando (oficialmente) días de ocho horas en el lado opuesto del reloj. Por lo que es probable que hubiese situaciones en las que cuando en la India llegaran a trabajar en la mañana, terminaran una tarea y necesitaran que la cola fuera resurtida, pero los directores de producto estaban obviamente dormidos en cama. Dado este problema de disponibilidad no inmediata, probablemente deberíamos de haber permitido 16 horas para aprovisionar un solo ítem en la cola bajo situaciones extremas. Recuerde que los desarrolladores eran el cuello de botella en este flujo de trabajo. Con el fin de maximizar rendimiento, nunca queremos que los desarrolladores estén ociosos. Por lo que necesitamos ser conservadores; 16 horas es conservativo cuando la decisión promedio de resurtido de la cola es de tan solo dos horas. Entonces, ¿cuál sería el rendimiento en un periodo promedio de 16 horas? En el rendimiento pico el equipo lograba 56 ítems en un trimestre. Eso es menos de cinco por semana. Por lo que en un periodo de 16 horas es poco probable que completen un solo ítem, es decir, un cola de tamaño uno es perfectamente aceptable. La carencia total de cola es inaceptable. Aún hay probabilidad de que el equipo sufra tiempo de ocio cuando terminen un ítem durante el lapso de 16 horas cuando los directores de producto no estén disponibles para aprovisionar la cola.

Secciones ilimitadas de flujo de trabajo

En la solución de sistema-de-arrastre de la Teoría de Restricciones para problemas de flujo, conocida como Drum-Buffer-Rope, todas las estaciones de trabajo en los niveles posteriores al cuello de botella tienen WIP ilimitado. Este diseño está basado en la asunción de que tienen una capacidad de mayor rendimiento que el cuello de botella y tienen capacidad de holgura que resulta en tiempo de ocio. Como resultado no hay necesidad de un límite de WIP. Esto está ilustrado en la Figura 10.2(a) basada en la metáfora usada en *The Goal* de Goldrat y muestra a un patrulla de scouts excursionando en una fila. La cuerda está atada entre el scout líder y el más lento (el cuarto en la línea), quien es el cuello de botella en el rendimiento (la tasa en la cual la patrulla cubre terreno). Se necesita tan solo una cuerda pues los scouts detrás del más lento nunca se van a retrasar porque pueden caminar más rápido que el cuarto en la línea, quién restringe el paso de la patrulla completa.

Figura 10.2 Hombres de palillo ilustrando cuatro diseños de sistemas de arrastre de WIP limitado

Con un sistema kanban, la mayoría o todas las estaciones en el flujo de trabajo tienen límites de WIP. Esta es una ventaja potencial porque los impedimentos causados por variabilidad no anticipada o esperada puede hacer que un paso en niveles altos del proceso se convierta en un cuello de botella temporal. El límite de WIP local con el sistema kanban detendrá la línea rápidamente, evitando que el sistema se obstruya y se sobrecargue. Cuando el impedimento es eliminado el sistema reinicia con gracia. La limitación de WIP estilo Kanban se ilustra en la figura 10.2(d), la cual muestra como los scouts estarán encordados juntos si usan un estilo de kanban. Es este caso cada scout está encordado al siguiente de la fila. Cinco cuerdas son requeridas para controlar el paso de la patrulla entera de seis scouts.

En algunos casos, puede ser aceptable que un sistema kanban tenga pasos ilimitados de nivel posterior del proceso. En el ejemplo del XIT en Microsoft se asumió que la base de usuario disponible para efectuar pruebas de aceptación era efectivamente infinita y en esencia disponible instantáneamente, por ello no había necesidad de limitar el WIP en las pruebas de aceptación de usuario. En Corbis, la cola de listo-para-entrega era ilimitada. Esto se basaba en la asunción de que el lote de trabajo listo-para-entrega nunca se haría excesivo dada la cadencia acordada de entregas cada dos semanas. Si, por el otro lado, hubiera sido posible que el monto de material listo-para-entrega se hiciera excesivo, habría incrementado la complejidad de la entrega, resultando en que el esfuerzo de coordinación y los costos de transacción de la entrega no fueran no económicos. Habría sido necesario limitar el WIP en la cola de listo-para-entrega. Sin embargo, ese nunca fue el caso en Corbis, por lo que el estado de listo-para-entrega se mantuvo sin restricción.

No estrese a su organización

Escoger límites de WIP demasiado ajustados puede poner estrés excesivo en su organización al principio. Organizaciones con poca madurez con capacidades pobres tendrán más impedimentos. Por ello, organizaciones de baja madurez y con poca capacidad se darán cuenta que la introducir un sistema kanban genera mucha inconveniencia si los límites de WIP son muy bajos. Si hay muchos impedimentos, representados con muchos tickets rosa a lo largo de la pared de tarjetas, límites de WIP demasiado ajustados significarán que todo llegará a un paro y mucha gente estará ociosa. Mientras que el tiempo de ocio tiende a enfocar la atención y acelerar los esfuerzos para resolver problemas y eliminar impedimentos, puede ser demasiado doloroso para una organización con poca madurez. La alta dirección pueden irritarse al observar a demasiada gente ociosa cobrando sus salarios.

Al introducir el cambio, usted necesita ser consciente del efecto de la curva “J”. Idealmente, cada cambio produce una pequeña “j” donde cualquier impacto en el rendimiento es superficial, el sistema se recupera rápidamente y muestra mejora. Si usted define los límites de WIP demasiado apretados sufrirá el efecto de una curva “J” demasiado pronunciada y larga, lo cual puede generar efectos indeseables: Kanban expone todos los problemas en la organización, pero puede terminar siendo acusada por hacer todo peor y será visto como parte del problema en lugar de como la solución. Actúe con cautela. Con organizaciones más capaces y maduras que sufren menores problemas inesperados (variaciones de causa assignable) usted puede ser más agresivo con sus políticas de límite de WIP. En organizaciones más caóticas es mejor introducir límites más holgados inicialmente, con mayor WIP y con la intención de reducirlos más adelante.

No establecer un límite de WIP es un error

Aunque le advierto que no sea agresivo al establecer los límites del WIP inicial, me he convencido que no establecer los límites de WIP es un error.

Algunos de los primeros en adoptar Kanban, como Yahoo!, optaron por no establecer límites de WIP porque asumían que sus equipos eran demasiado caóticos para sobrellevar el dolor que puede introducir. La esperanza era que estas organizaciones maduran a través de los elementos visuales de control de Kanban y que los límites de WIP podrían introducirse más tarde. Esto resultó ser problemático, sin embargo y varios equipos abandonaron Kanban sin ver grandes mejoras, mientras que otros fueron disueltos en reorganizaciones corporativas o por cancelaciones de proyectos, negándose a dar datos adicionales. En Corbis, varios equipos en grandes proyectos usaban Kanban con límites de WIP muy holgados en la funcionalidad de alto nivel refinada conforme se avanzaba. Los resultados fueron mixtos en cierta medida.

Me he convencido de que la tensión creada por la imposición de un límite de WIP a través de la cadena de valor es tensión positiva. La tensión positiva fuerza discusión sobre las disfunciones y problemas de la organización. Las disfunciones generan impedimentos en el flujo y resultan en productividad, tiempo de entrega y calidad subóptimas. La discusión y colaboración provocada por la tensión positiva de un límite de WIP es saludable. Es el mecanismo que permite el surgimiento de una cultura de mejora continua. Sin límites de WIP el progreso en la mejora del proceso es lento. Los equipos que han impuesto límites de WIP desde el principio han reportado crecimiento acelerado en su capacidad y madurez organizacional y han entregado resultados de negocio superiores con entregas previsibles frecuentes de software de alta calidad. En comparación, los equipos que han aplazado la introducción de límites de WIP han luchado en general y mostrado mejoras limitadas.

Asignación de la capacidad

Una vez que establecemos los límites de WIP para el flujo a través del sistema, podemos considerar la asignación de la capacidad por tipo de ítem de trabajo o clase de servicio.

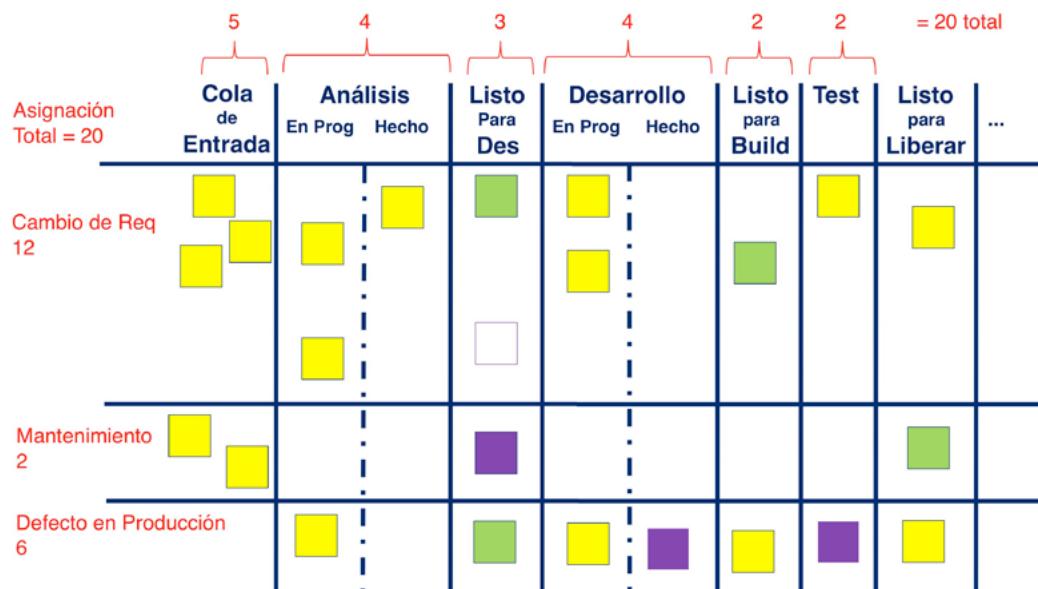


Figura 10.3 Pared de tarjetas mostrando carriles de nado para cada tipo de ítem de trabajo con límites de WIP explícitos para cada carril

La Figura 10.3 muestra el diseño de la pared de tarjetas del capítulo 6 con límites de WIP a través de columnas totalizando 20 tarjetas. La capacidad es asignada a través de los tipos de ítem de trabajo, es decir 60 por ciento para solicitudes de cambio, 10% para ítems

de mantenimiento y 30% para cambios de texto en producción. Eso equivale a un límite de WIP por carril de nado de 12 para solicitudes de cambio, 2 para mantenimiento y 6 para cambios de texto en producción.

La asignación de capacidad nos permite garantizar servicio para cada tipo de trabajo recibido por el sistema kanban. La asignación debe hacerse generalmente en respuesta a una demanda comparativa observada para cada tipo de trabajo. Por ende, es importante completar un análisis de demanda para facilitar una asignación razonable de límites de WIP en los carriles de nado para cada tipo de trabajo.

Para llevar

- ❖ Los límites WIP deben ser acordados a través de consenso con las partes interesadas de los niveles anteriores y posteriores del proceso y con la alta dirección funcional.
- ❖ La declaración unilateral de límites de WIP es posible pero pueden ser difíciles de defender más tarde, cuando los sistemas están bajo estrés.
- ❖ Los límites de WIP para tareas de trabajo deben establecerse como un numero promedio de ítems por persona, par de desarrollo, o equipo colaborativo pequeño.
- ❖ Normalmente , el límite debe estar dentro del rango de uno a tres ítems por persona, par, o equipo.
- ❖ Los límites de la cola deben mantenerse pequeños, habitualmente los suficientemente largos para absorber la variación natural (causa-aleatoria) en el tamaño de los ítems y la duración de la tarea.
- ❖ Los cuellos de botella deben tener buffer.
- ❖ El tamaño de los buffers debe ser lo más pequeño posible, pero lo suficientemente largo para asegurar rendimiento óptimo en el cuello de botella y el mantenimiento de flujo en el sistema.
- ❖ Todos los límites de WIP pueden ajustarse empíricamente.
- ❖ Kanban es un proceso empírico.
- ❖ No debe desperdiciarse demasiado tiempo intentando determinar el límite de WIP perfecto; simplemente elija un número que es lo suficientemente cercano y progrese. Ajústelo empíricamente de ser necesario.
- ❖ Las secciones ilimitadas de niveles posteriores del proceso son posible.
- ❖ Debe tenerse cuidado de que el establecimiento de flujo de trabajo ilimitado no introduzca cuellos de botella o cause costos excesivos de transacción o coordinación cuando se hacen entregas (transferencias de lotes a niveles posteriores de proceso).
- ❖ Una vez que los límites de WIP han sido establecidos, la capacidad puede ser asignada a través de los tipos de ítems de trabajo.
- ❖ Los carriles de nado son utilizados frecuentemente para cada tipo de ítem de trabajo y un límite de WIP es establecido para cada carril.
- ❖ La asignación de capacidad requiere un análisis comparativo de demanda a través de los tipos distintos de trabajo recibido por un sistema kanban.

❖ CAPÍTULO 11 ❖

Estableciendo los límites del Trabajo-en-Progreso

Estamos familiarizados con el concepto de diferentes clases de servicio. Cualquiera que se ha registrado para un vuelo en un aeropuerto entiende que los clientes que pagan más por su boleto, o quienes disfrutan de los beneficios de un programa de lealtad del cliente, tienen permitido usar una línea rápida para “cortar la cola”. A veces estos privilegios se extienden a líneas de seguridad de los aeropuertos e incluyen el uso de salas exclusivas o tratamiento especial para embarcar. Clientes que pagan más o gastan más dinero en la aerolínea regularmente disfrutan de una mejor clase de servicio.

Estamos familiarizados también con este concepto en el desarrollo de software y en el trabajo de sistemas de TI, más evidentemente con la corrección de defectos y particularmente con defectos en producción. Evaluamos defectos por severidad (impacto) y prioridad (urgencia). Defectos con alta severidad y prioridad son resueltos inmediatamente. Reciben una clase de servicio diferente y más alta que otro trabajo. Para resolver un defecto de alta severidad en producción ponemos el trabajo actual a un lado, tomamos tanta gente como sea necesario e inmediatamente hacemos planes especiales para una reparación de emergencia, parche, o entrega para aliviar el problema.

Este concepto puede aplicarse más generalmente, lo cual ofrece algunas ventajas tanto en la agilidad de negocio como en la administración de riesgos. Algunas solicitudes se necesitan más rápidamente que otras mientras que algunas son más valiosas que otras. Ofreciendo tratar

distintos tipos de trabajo con diferentes clases de servicio podemos ofrecerle al cliente más flexibilidad mientras que optimizamos el resultado económico.

Las clases de servicio nos dan una forma conveniente de clasificar el trabajo para brindar niveles aceptables de satisfacción del cliente a un costo económico óptimo. Identificando la clase de servicio de un ítem rápidamente nos ahorra la necesidad de hacer una estimación detallada o análisis. Políticas asociadas con una clase de servicio afectan la manera como los ítems son tomados a través del sistema. La clase de servicio determina la prioridad dentro del sistema. Las clases de servicio permiten un enfoque auto-organizado, de valor y riesgo optimizado para priorización y “replantificar”.

Definiciones de las clases de servicio habituales

Las clases de servicio están normalmente definidas en base a su impacto de negocio. Se asignan tarjetas adhesivas de distintos colores, tarjetas de índice, o tickets a cada clase y claramente señalan la clase de servicio de una solicitud dada, como se muestra en la Figura 11.1; de lo contrario se dibujan carriles de nado en la pared de tarjetas para significar membresía en una clase de servicio.

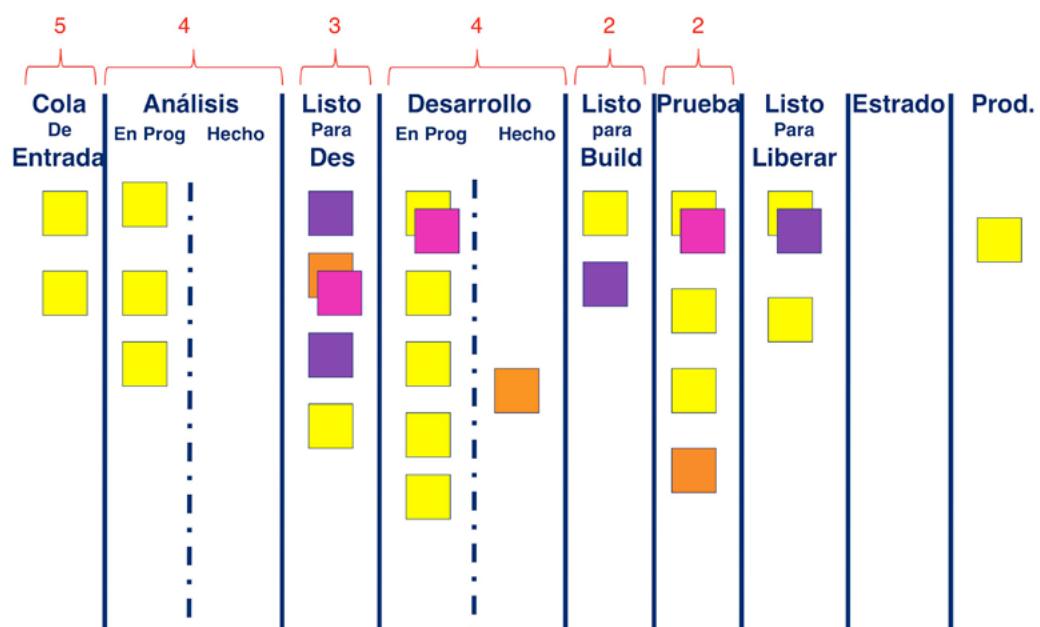


Figura 11.1 Pared de tarjetas mostrando tickets de colores para señalar las clases de servicio

Cada clase de servicio viene con su propio conjunto de políticas que determinan cómo los ítems son priorizados cuando son llevados a través del sistema kanban. La clase de

servicio también incluye un compromiso explícito para el cliente. A continuación se da un ejemplo breve de un conjunto de definiciones de clase de servicio. Mientras que este conjunto no es una copia exacta de ninguno utilizado en alguna implementación específica Kanban, representa una fuerte generalización de las clases de servicio observadas en el campo.

En este conjunto de ejemplos se ofrecen cuatro clases de servicio. Como guía general, es recomendable ofrecer hasta un máximo de seis clases. Demasiadas serán muy difíciles de administrar y operar. El número de clases de servicio deben ser lo suficientemente pocas como para que todos los miembros involucrados—miembros del equipo y partes interesadas externas—puedan recordarlas todas, mientras que son suficientes como para ofrecer flexibilidad para responder a la demanda del cliente.

Expreso

La clase de servicio Expreso (o “Bala de Plata”) está bien entendida en la industria de manufactura. Un escenario habitual podría ser un equipo de venta intentando lograr un objetivo de ventas trimestrales en acoplamiento con un cliente que tiene un presupuesto que debe ser invertido antes del término del año fiscal. El cliente ha estado retrasando la decisión de compra; finalmente toma una decisión justo cuando el tiempo se agote en el año fiscal en curso. El pedido se hace pero está condicionado a la su entrega antes de la fecha límite. El fabricante está de acuerdo con el precio y la cantidad y acepta el pedido, el cual debe cumplirse, entregarse y facturarse antes del último día del trimestre. El pedido normalmente llega a la fábrica a través de la oficina del vicepresidente de ventas regionales con una solicitud para acelerar la entrega dado lo apretado del calendario y el valor del pedido.



La capacidad para acelerar le da al vendedor la capacidad de decir “¡Sí!” bajo circunstancias difíciles para satisfacer la necesidad del cliente. Sin embargo, los pedidos expreso afectan mucho a las cadenas de suministro de fabricación y distribución. Expreso es conocido en ingeniería industrial e investigación de operaciones tanto para aumentar los niveles de inventario como para aumentar los tiempos de entrega de otros pedidos que no sean expresos. El área de negocio tiene la opción de obtener un valor en una venta específica al costo de retrasar otras órdenes y de incurrir en costos adicionales de niveles más altos de inventario. Si la empresa está bien dirigida, el valor generado por agilizar superará los costes debidos a plazos más largos (y potenciales perdidas de negocio como consecuencia) y al costo de elevar el nivel de inventario a uno más alto.

Las empresas de manufactura a menudo crean políticas para limitar el número de solicitudes expreso que llegan a la fábrica. Una política común es el otorgar un número fijo de las llamadas “balas de plata” a un vicepresidente de ventas en un periodo de tiempo

dado. Por ello el término “bala de plata” se ha hecho sinónimo de hacer algo expreso en la manufactura o distribución.

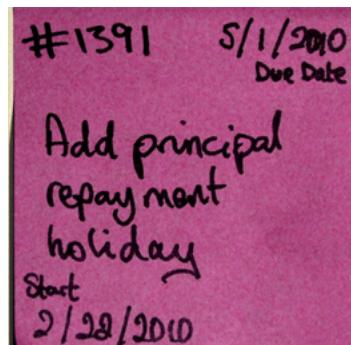
Desafortunadamente, por claridad, el término “bala de plata” estaba ya en uso en la Ingeniería de Software. Fred Brooks lo definió como un solo cambio (en tecnología o proceso) que crearía una mejora de una orden de magnitud (diez veces) en la productividad del programador. Por eso, recomiendo que se apegue al término Expreso para esta clase de servicio. En compañías que hacen manufactura, o donde la alta dirección está familiarizada con manufactura, sin embargo, he observado que ellos prefieren el término “bala de plata”. Esto está bien mientras la gente de tecnología entienda la diferencia de uso.

Fecha de entrega fija

A mediados de febrero de 2007, un desarrollador entró en mi oficina para preguntarme si era consiente de un asunto con una plataforma de servicio que usamos para procesamiento de tarjetas de crédito. No lo estaba, por lo que él me lo explicó. Aparentemente, el vendedor encontró que código base era muy difícil de mantener cuando intentaron agregar más y más características a su plataforma—un mal común en el desarrollo de software. Con el fin de conseguir las solicitudes de nuevas características en 2006, habían reemplazado su plataforma completa con un nuevo sistema que tenía una interface de programación de aplicación (API). Informaron a todos sus clientes y les notificaron que en un período de 15 meses el sistema antiguo sería eliminado después del 31 de marzo de 2007. Poniéndolo de otra manera, si no actualizábamos nuestros sistemas para usar la nueva plataforma nuestra actividad comercial en el Internet cesaría el 1 de abril de 2007. Esto sería un gran inconveniente para un negocio que hacía muchos de sus ingresos mediante ventas basadas en la web, sin mencionar lo vergonzoso que sería para el propietario de la firma. Contábamos con seis meses para hacer los cambios necesarios y entregar el nuevo código a producción. El ticket para este trabajo entró a nuestro sistema kanban marcado con una fecha fija de terminación. La información adicional en el ticket tenía la intención de atraer la atención hacia el costo y el impacto de una entrega tardía, así como permitirle al equipo que hiciera el ítem expreso y asegurara la entrega a tiempo.

No era la primera vez que veíamos tal tipo de solicitud. Existía una solicitud anterior relacionada con la integración de los sistemas de TI de una empresa adquirida. La fecha “fija” se había derivado del caso de negocio para la adquisición, la cual mostraba ahorros significativos de costo desde febrero de ese mismo año.

Un tema o patrón parecía estar emergiendo. Había algunas solicitudes relacionadas a obligaciones contractuales mayores, algunas a requerimientos regulatorios (usualmente del gobierno federal) y algunas a iniciativas estratégicas, tales como la adquisición de otro



negocio. Solicitudes de esta naturaleza acarreaban un costo de retraso significativo, directo o indirecto, que tendía a caer dentro de una de dos categorías: habría una fecha en la que se generaría una multa—un costo directo, específico, de bolsillo—impuesta por una autoridad regulatoria o por los términos establecidos en el contrato. Alternativamente, habría un requerimiento para detener alguna actividad, tal como la venta de un determinado tipo de elemento u operar en una determinada jurisdicción, hasta que los requisitos se cumplieran. Este segundo costo indirecto es un costo de oportunidad perdida—los ingresos potenciales perdidos durante el período de retraso. Ambos tipos se muestran en la figura 11.2. Todo aquello que tiene una “ventana de entrega” física o cultural debe ser considerada como si tuviera una fecha de entrega fija, si esa fecha futura cae dentro de una ventana de tiempo de entrega razonable a partir del tiempo presente.

Los negocios con calendarios estacionales, como escuelas y colegios, tienden a tener limitaciones de calendario. Si está trabajando en un sector tal como el de educación, puede tener clientes que desean la entrega del software en fechas fijas del año o de lo contrario no los desearán en absoluto: La falta de lanzar dentro de ese plazo resulta en una venta perdida. Todo lo que tiene un plazo físico debería ser considerado como algo que tiene una función de costo de retraso y debe ser tratado como si hubiera una fecha de entrega fija.

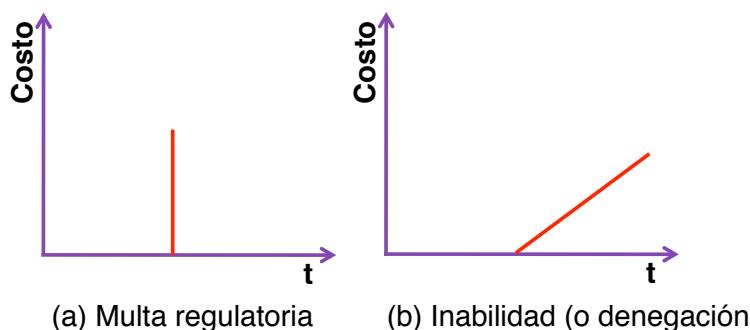


Figura 11.2 Dos perfiles de la función de coso de retraso para la clase de servicio de Fecha Fija

Clase estándar

La mayoría de los ítems que se necesitan con alguna urgencia deberían ser tratados como ítems de clase estándar. Las políticas y acuerdos de nivel de servicio para una clase estándar pueden variar por el tipo de ítem de trabajo. Un esquema de diseño de sistema kanban común separa los tipos de trabajo por tamaño, tales como pequeño, mediano y grande.

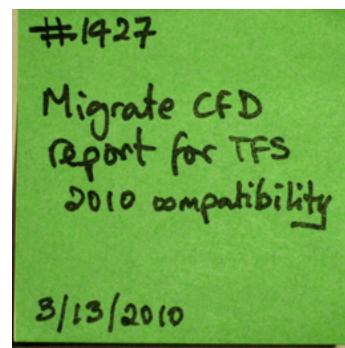


Se puede ofrecer un acuerdo de nivel de servicio diferente para ítems de clase estándar de cada tamaño. Por ejemplo, ítems pequeños son procesados habitualmente dentro de cuatro días, ítems de tamaño medio dentro de un mes, e ítems grandes dentro de un trimestre. Los ítems de clase estándar tienden a tener un costo tangible de retraso que puede ser calculado (aunque no necesariamente en unidades monetarias). El costo de retraso tenderá a suceder pronto—dentro del plazo de la entrega de la solicitud. El costo de retraso tiende a ser inmediato: ¡si tuviéramos esta función hoy, el beneficio se derivaría mañana!

Clase intangible

Tiene sentido ofrecer una cuarta, y más baja, clase de servicio. He luchado por encontrar un nombre adecuado para esta clase y me decidí por “intangible”. No estoy enteramente contento con él por lo que puede que lo cambie en una edición futura del libro. Los ítems de clase intangible pueden ser importantes y valiosos, pero no hay costo de retraso tangible asociado con ellos en el futuro cercano. Es decir, no hay costo de retraso dentro del plazo que puede tomar entregar el ítem. Las solicitudes que se ajustan a este patrón se relacionan con resultados con fechas potencialmente fijas que están muy lejos en el futuro, tales como reemplazo de plataforma.

Por ejemplo, en 2005 Microsoft lanzó SQL Server 2005, la última versión de su servidor RDBMS de base de datos. La edición 2005 reemplazó la edición 2000 y se convirtió en el “final de la línea”. Como el jugador dominante en el mercado, Microsoft está obligado a soportar sus productos por diez años después de ser introducidos. Como resultado, el soporte para el SQL Server 2000 necesita continuar hasta 2010. Esto provee a los clientes con un plazo de 5 años para reemplazar código que es incompatible con las ediciones, 2005 o 2010, más nuevas de la plataforma. Por lo que en 2005 ó 2006, reemplazar el código de la base de datos—*store procedures*, código de persistencia—no es una actividad de prioridad crítica. No se incurre en costo de retraso durante esos años. Sin embargo, conforme el tiempo pasa y el código no es cambiado, el costo se incrementa. Se hace incrementalmente más difícil trabajar con otros productos conforme nuevas versiones requieren SQL Server 2005 como un requisito previo. Más y más presión se acumula para hacer el cambio a la nueva plataforma. Para 2009 el asunto es urgente pues Microsoft cesará el soporte para el producto viejo y la falla de actualización hará que el negocio corra máquinas viejas con sistemas operativos viejos e infraestructura asociada no soportados. Si el riesgo es inaceptable, el código debe ser actualizado. Este problema de reemplazo de plataforma es un problema común con el que los equipos de ingeniería de software luchan continuamente. Existe el deseo de iniciar el trabajo temprano y completarlo a tiempo, pero la capacidad para hacer el trabajo de actualización es desplazado por otro trabajo que es más urgente o crítico. En otras palabras, el



reemplazo de plataforma, aunque tiene un costo inmediato bajo de retraso, es reemplazado por otro trabajo con costo de retraso mayor y más inmediato.

Tiene sentido ofrecer una clase de servicio que permita que tal tipo de trabajo sea atendido pronto. La capacidad se puede asignar para asegurar que el trabajo es completado; sin embargo, no debe haber garantía de tiempo. En adición, es este trabajo de bajo costo de retraso lo que siempre se pondrá a un lado cuando lleguen solicitudes más urgentes . A fin de tener holgura para procesar una solicitud Expreso, debe haber trabajo de bajo costo de retraso que puede ser desplazado mientras que la solicitud Expreso es procesada. Estos ítems intangibles proveen la holgura.

Políticas para las clases de servicio

Debe emplearse una técnica de visualización para identificar la clase de servicio fácilmente. Como se mencionó anteriormente, lo más común es utilizar tickets de distintos colores o carriles de nado en la pared de tarjetas. Algunos equipos han utilizado decoraciones tales como calcomanías pegadas en un ítem de trabajo. Un carril de nado para solicitudes expresivo también es una elección común. La manera de visualizar las clases de servicio depende de usted mismo. Este capítulo utiliza distintos colores para significar las clases de servicio. El objetivo es asegurarse de que cualquier persona involucrada, en cualquier día, pueda utilizar las políticas de priorización simples asociadas con una clase de servicio para tomar decisiones de priorización de buena calidad en el campo sin intervención o supervisión gerencial.

A continuación hay un ejemplo de políticas de priorización para las cuatro clases de servicio definidas previamente. Naturalmente, las definiciones de clase de servicio serán únicas y sus políticas de uso serán diferentes de estos ejemplos en cada implementación. Estas políticas están basadas en evidencia empírica y reflejan de manera bastante precisa políticas que equipos reales han utilizado.

Políticas expresivo

- Las solicitudes expresivo utilizan tarjetas blancas.
- Solamente una solicitud expresivo es permitida en un momento dado. En otras palabras, la clase de servicio expresivo tiene un límite de WIP de 1.
- Un recurso calificado debe tomar solicitudes expresivo inmediatamente. Cualquier otro trabajo se pone en espera para poder procesar la solicitud expresivo.
- En cualquier punto en el flujo de trabajo, el límite puede ser excedido con el fin de acomodar la solicitud expresivo. La capacidad no se mantiene en reserva para una tarea expresivo.

- En cualquier punto en el flujo de trabajo, el límite Kanban puede ser excedido con el fin de acomodar la solicitud expreso. La capacidad no se mantiene en reserva para una tarea expreso.
- De ser necesario, una entrega especial (fuera de ciclo) será planeada para poner una solicitud expreso en producción lo más temprano posible.

Políticas de fecha de entrega fija

- Los ítems de fecha de entrega fija utilizan tarjetas púrpura.
- La fecha de entrega requerida se muestra en la parte inferior derecha de la tarjeta.
- Los ítems de entrega de fecha fija reciben algún análisis y una estimación de tamaño y esfuerzo puede llevarse a cabo para determinar el tiempo de flujo. Si el ítem es grande puede romperse en ítems más pequeños; cada ítem pequeño será evaluado independientemente para ver si califica como un ítem de fecha de entrega fija.
- Los ítems de fecha de entrega fija permanecen en el *backlog* hasta que son seleccionados para la cola de entrada, cerca del momento ideal en el que pueden ser entregados a tiempo dada la estimación del flujo-tiempo.
- Los ítems de fecha de entrega fija son tomados preferentemente sobre otros ítems de menos riesgo. En este ejemplo son tomados antes que los ítems de clase estándar o intangibles.
- Los ítems de fecha de entrega fija deben adherirse al límite de WIP.
- Una vez completos y listos para entrega, los ítems de fecha de entrega fija se ponen en la cola de entrega. Son lanzados en una entrega programada regularmente en fecha previa a la requerida de entrega.
- Si un ítem de fecha de entrega fija se retrasa y la entrega en la fecha deseada está bajo riesgo, su clase de servicio puede ser promovida a solicitud expreso.

Políticas de clase estándar

- Ítems de clase estándar utilizan tarjetas amarillas.
- Los ítems de clase estándar son priorizados y puestos dentro de la cola de entrada en base a un mecanismo acordado, tal como voto democrático y habitualmente son seleccionados basándose en su coto de retraso o valor de negocio.
- Los ítems de clase estándar usan la técnica de cola de primero-en-entrar-primeroen-salir (FIFO) conforme son arrastrados a través del sistema. Normalmente y

cuento se le da la opción, un miembro del equipo toma el ítem de clase estándar más viejo si no hay un ítem expreso o de fecha fija por escoger preferentemente.

- Una vez completos y listos para entrega, los ítems de clase estándar fija se ponen en la cola de entrega. Son lanzados en la siguiente entrega programada.
- No se efectúa ninguna estimación para determinar el nivel de esfuerzo o tiempo de flujo.
- Los ítems de clase estándar pueden ser analizados en cuanto al orden de magnitud en tamaño, habitualmente pequeño (unos cuantos días), medio (una o dos semanas), y grande (tal vez meses).
- Los ítems de clase estándar son generalmente entregados dentro de x días de su selección con un rendimiento de fecha límite del m por ciento.

El acuerdo habitual de nivel de servicio para una clase estándar puede ofrecer un tiempo de entrega de 30 días con rendimiento de fecha límite del 80 por ciento. En otras palabras, cuatro de cada cinco solicitudes deberían ser entregadas dentro de 30 días.

Clase intangible

- Ítems de clase intangible utilizan tarjetas verdes.
- Los ítems de clase intangible son priorizados y puestos en la cola de entrada en base a un mecanismo acordado, tal como voto democrático y normalmente son seleccionados en base a algún impacto de largo plazo o costo de retraso.
- Los ítems de clase intangible son tomados a través del sistema de manera ad hoc. Los miembros del equipo pueden elegir tomar un ítem de clase intangible independientemente de su fecha de entrada, siempre y cuando un ítem de clase más alta no esté disponible.
- Una vez completos y listos para entrega, los ítems de clase intangible fija se ponen en la cola de entrega. Son lanzados en la siguiente entrega programada o hasta que son ensamblados con otros ítems.
- No se efectúa ninguna estimación para determinar el nivel de esfuerzo o tiempo de flujo.
- Los ítems de clase intangible pueden ser analizados en cuanto a tamaño. Ítems largos pueden ser separados en ítems más pequeños. Cada ítem puede ser encolado y puesto en flujo separadamente.
- Normalmente , un ítem de clase intangible es puesto a un lado para poder procesar una solicitud expreso.

Puede que no sea necesario ofrecer un acuerdo de clase de servicio con ítems de clase intangible. Si lo es entonces debe ser un acuerdo significativamente más holgado que el ofrecido para los ítems de clase estándar; por ejemplo, 60 días con 50% de rendimiento de fecha límite.

Determinando un objetivo para la entrega de servicio

En el ejemplo dado hace un momento sobre las clases de servicio, la clase de servicio Estándar utiliza un tiempo de entrega límite; por ejemplo, 28 días (cuatro semanas). El concepto de ofrecer un objetivo de tiempo de entrega acoplado con una métrica de rendimiento de fecha límite es una alternativa para tratar cada ítem individualmente y tener que estimar y comprometerse a una fecha de entrega para cada ítem. El acuerdo de nivel de servicio nos permite evitar actividades costosas tales como la estimación; actividades de poca confianza tales como comprometerse; y difundir riesgo agregando una larga colección de solicitudes y prometer rendimiento agregado en la forma de un porcentaje de ejecución con fecha límite. Evitando hacer promesas poco probables de cumplir, evitamos el peligro de perder la confianza de nuestros clientes. Por lo tanto, es importante comunicar que el tiempo de entrega límite en la clase de servicio Estándar es tan solo eso, un objetivo.

Para determinar el tiempo de entrega límite, es útil tener algunos datos históricos. Si no tiene ninguno, elucubre razonablemente. Si cuenta con ellos entonces la manera más científica para determinar el tiempo de entrega límite es procesar los tiempos de entrega (desde la primer selección hasta la entrega) a través de un paquete estadístico de control de procesos, o una herramienta de seguimiento de kanban que soporte control estadístico de proceso (tal como Silver Catalyst) y usar el límite superior de control como su tiempo de entrega. Esto asegura un tiempo que puede lograr bajo la mayoría de las circunstancias normales y no logarlo únicamente cuando hay un problema único de causa-asignable (ver el capítulo 19 para una explicación más detallada).

Si, por el otro lado, el último párrafo no significó nada para usted, entonces una explicación más somera es que usted desea que el tiempo de entrega sea alcanzable la mayoría de las veces, pero también que sea lo suficientemente agresivo para mantener al equipo enfocado. Es probable que sus ítems de trabajo varíen en tamaño, complejidad, riesgo y experiencia requerida. De ser así, los tiempos de entrega varían significativamente. Eso está bien. Si efectúa un análisis espectral sobre algunos datos históricos y puede ver que tal vez el 70% son entregados dentro de 28 días y el 30% sobrante se esparce sobre otros 100 días, entonces puede ser razonable sugerir un tiempo de entrega límite de 28 días.

He aprendido que el uso de clases de servicio es una técnica muy poderosa. Con mi equipo en 2007, aproximadamente el 30% de todas las solicitudes se tardaban comparado con el tiempo de entrega límite. Reportamos esto como la métrica de Rendimiento de Fecha Límite. Nunca fue mayor al 70%. Sin embargo y a pesar de este lamentable desempeño

contra la fecha objetivo, teníamos muy pocas quejas. La razón de esto se hizo evidente: todos los ítems importantes—aquejellos de alto riesgo o alto valor—siempre estaban listos a tiempo y había confianza de que los tardíos serían entregados dentro de dos a cuatro semanas adicionales, conforme las entregas sucedían a intervalos regulares de manera confiable.

Las clases de servicio Expreso y Fecha de Entrega Fija aseguraban que ítems importantes siempre estaban listos a tiempo. Mientras tanto, la otros ítems de clase Estándar que estaban retrasados lo estaban por uno o dos entregas (14 o 28 días respectivamente) generalmente. Los clientes confiaban en la cadencia de entrega. Esa confianza era ganada haciendo las cosas. Consistentemente enviábamos una entrega cada segundo miércoles. Con el costo insignificante de retraso asociado con muchos ítems de clase Estándar (e Intangible), el negocio se enfocaba en lo que había sido entregado y en planificar para futuros ítems en lugar de preocuparse mucho sobre fechas de entrega precisas del trabajo-en-progreso.

El resultado era significativo porque Kanban con clases de servicio había cambiado claramente la psicología del cliente y la naturaleza de la relación y las expectativas. Los clientes se orientaban hacia la relación a largo plazo y el rendimiento del sistema y no en la entrega de ítem o ítems específicos. Esto le dio al equipo de desarrollo la libertad de enfocarse en lo correcto y no perder tiempo atendiendo asuntos que surgían del poco nivel de confianza entre ellos y sus clientes.

Asignando una clase de servicio

La clase de servicio para un ítem debe ser asignada cuando el ítem es seleccionado en la cola de trabajo. Debe ser auto-evidente si el ítem es una solicitud Expreso—viene con una solicitud de que el ítem sea procesado lo más pronto posible. Esto se justifica basándose en un caso de negocio que permite una oportunidad inmediata o identifica un costo significativo en que se incurrirá si la solicitud no es satisfecha. Tal vez ya se está incurriendo en ese costo. Esto es habitual con defectos en producción de alta severidad.

También debe ser auto-evidente por naturaleza si un ítem tiene una fecha de entrega fija. Tal vez la solicitud se relaciona a requerimientos normativos nuevos establecidos por una autoridad regulatoria independiente, o algo de carácter estacional del negocio. Si un ítem es de clase de fecha fija entonces la fecha es habitualmente conocida y cae dentro de un marco de tiempo razonable—tal vez el doble de largo comparado con el tiempo de entrega objetivo de la clase de servicio estándar típica—y el ítem puede ser estimado como para ser introducido en el punto óptimo para asegurar su entrega a tiempo.

Una elección más difícil está en el hecho de si un ítem es de clase estándar o intangible. Mi observación es que ítems de clase estándar habitualmente tienen funciones de oportunidad-costo que tienen efecto inmediatamente. Por ejemplo, si tuviéramos esta nueva característica hoy podríamos estar haciendo dinero de él desde mañana. Por lo que una

entrega pronta es deseable, pero el retraso no tiene las mismas consecuencias negativas comparada con algo de fecha fija o una solicitud expreso.

Ítems intangibles tienden a estar asociados con ítems valiosos e importantes que tienen un costo (oportunidad) de retraso que no tiene efecto en el futuro cercano. Normalmente , el punto en el cual la función de costo se hace un punto de infección hacia arriba son trimestres o años en el futuro. Está más allá del horizonte de planificación inmediato, e. g., dos o tres veces el tiempo de entrega habitual. Si nuestro tiempo de entrega actual es generalmente de 28 días, entonces nuestro horizonte de planificación es probablemente tres meses. Ítems que incurren en una oportunidad perdida o en un costo tangible dentro de una ventana de tres meses deberían ser tratados como de clase estándar, mientras que ítems en los que el costo o beneficio no se produce sino hasta trimestres o años en el futuro deberían ser tratados como ítems de clase intangible.

Poniendo en uso las clases de servicio

Las clases de servicio deben definirse para cada sistema kanban. Las políticas asociadas con cada clase de servicio deben ser explicadas a cada miembro del equipo. Todos los que atienden la reunión de pie en la mañana deben apreciar y entender las clases de servicio en uso. Para hacer esto efectivo, el número de clases de servicio debe mantenerse bastante pequeño—cuatro o seis es una buena guía. Nuevamente, debido a que deseamos que cada miembro del equipo recuerde las clases de servicio, lo que significan y cómo usarlas, el número de políticas para cada clase de servicio debe mantenerse pequeña y simple. Las definiciones deben ser precisas y no ambiguas. Es decir, una buena guía debe ser de no más de seis políticas por clase de servicio.

Armado con un entendimiento de las clases de servicio y conocimiento de las políticas asociadas con ellos, el equipo debe estar apoderado para auto-organizar el flujo de trabajo. Los ítems de trabajo deben fluir a través del sistema de manera tal que optimice el valor del negocio y el servicio al cliente, resultando en entregas de software que maximicen la satisfacción del cliente.

Asigne capacidad a las clases de servicio

La Figura 11.3 muestra un sistema kanban con un límite de total de 20. Las clases de servicio están designadas en tickets de cuatro colores. Los tickets expreso blancos no cuentan en el límite de WIP, pero están limitados a tan solo un ítem a la vez. Por ende, tienen un 5% de impacto en la capacidad total cuando están presentes e incrementan el trabajo-en-progreso efectivo a 21 ítems. Es este ejemplo, los tickets púrpuras de fecha de entrega fija representan 20% del total. Esto significa que solo puede haber cuatro tickets púrpuras en

el tablero en un momento dado, pero pueden estar presentes en cualquier columna. Los ítems de clase estándar amarillos representan el 50% de la asignación total, para un total de diez tickets. El 30% sobrante es asignado a ítems de clase intangible verdes.

Ya que hemos asignado la capacidad a las diferentes clases de servicio, la actividad de aprovisionar la cola de entrada se complica debido a la capacidad disponible de cada clase. Como se muestra actualmente, hay capacidad para un ítem de fecha de entrega fija y tres ítems de clase intangible. Esto genera muchas preguntas. ¿Qué sucede si no tenemos demanda actual para un ítem de fecha fija? ¿Qué hacemos? ¿Llenamos el espacio con un ítem de clase estándar? Si es así, ¿debe el estado asignado a ese ítem ser de fecha fija o tratado como ítem de clase estándar? Si hacemos eso, ¿no se estaría ajustando la política de asignación de capacidad?

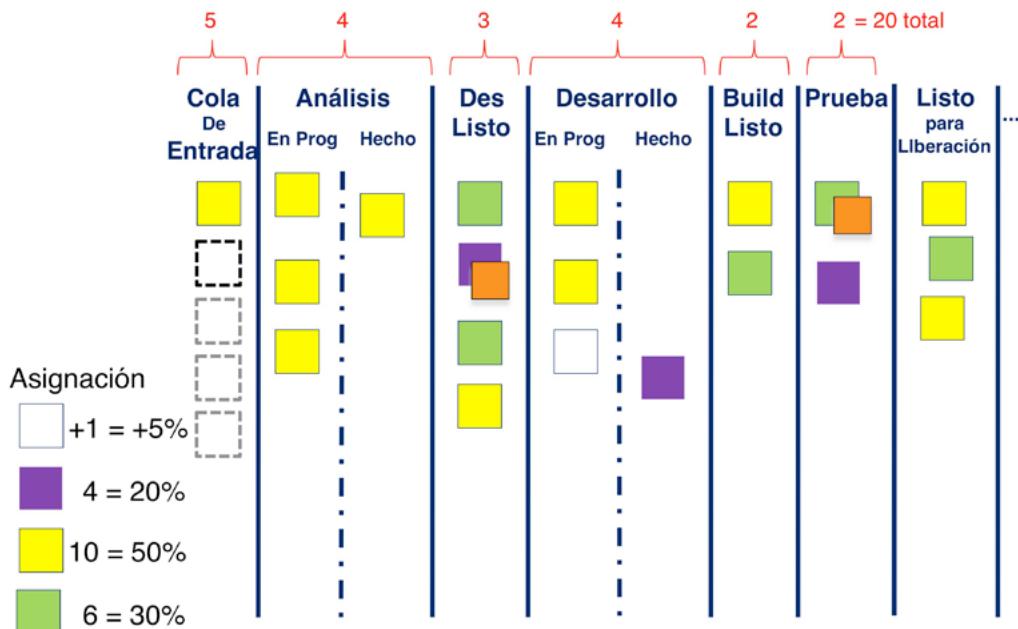


Figura 11.3 Pared de tarjetas mostrando la asignación de capacidad a través de las clases de servicio

Todas ellas son preguntas legítimas sobre asuntos legítimos que surgen diariamente cuando se usa un sistema kanban. No hay respuestas buenas o malas a esas preguntas. Las respuestas deben derivarse del contexto y son específicas a cada situación.

Lo que podemos deducir de la asignación escogida es que el dominio tiene un número significativo de ítems de naturaleza de fecha fija y que también hay una capacidad medible puesta aparte para ítems intangibles. Esto puede implicar que hay algunas iniciativas mayores con fechas de entrega más largas—tales como reemplazo de plataformas—en camino. Puede indicar también riesgos significativos en el dominio. Probablemente hay una

naturaleza estacional en la demanda que incrementa el número de solicitudes expreso o ítems de fecha fija. Con el fin de poder responder a esta demanda estacional adecuadamente y sin generar más insatisfacción del cliente, elegimos asignar más capacidad a ítems intangibles en lugar de a ítems de clase estándar. Estamos dándole más holgura al sistema.

En cuanto a elecciones sobre qué hacer cuando nuestra cola de entrada tiene un espacio de fecha fija disponible cuando no hay ítems de fecha fija adecuados disponibles; esto depende de los riesgos presentes en el dominio. Si hay demanda significativa por ítems de fecha fija y los costos asociados con estos ítems son altos (por lo que el riesgo es alto), podemos escoger dejar el espacio vacío. Puede tener sentido reservar la capacidad para un ítem de fecha de entrega fija en el futuro. Sin embargo, si los riesgos son bajos, podemos escoger llenar el espacio con un ítem de clase estándar. Si un ítem de fecha fija llega más tarde, podríamos escoger bajar el ítem de clase estándar a una clase más baja o exceder el límite de WIP temporalmente. Todas estas elecciones tendrán diferente efecto en el rendimiento del tiempo de entrega, dispersión de variabilidad en el tiempo de entrega, satisfacción del cliente y gestión de riesgo. Usted necesita hacer estas decisiones por usted mismo. Tomará algún tiempo desarrollar la experiencia y juicio adecuados para poder hacer las mejores elecciones para su equipo, proyecto, u organización.

La asignación de capacidad es tan solo una estrategia más del sistema kanban. Si usted se da cuenta que su asignación no está alineada con la demanda entonces ajústela—cambie las políticas y ajuste los límites de WIP de acuerdo a ellas.

Para llevar

- ❖ Las clases de servicio ofrecen un método abreviado para optimizar la satisfacción del cliente.
- ❖ Los ítems de trabajo deben ser asignados a una clase de servicio en función de su impacto en el negocio.
- ❖ Las clases de servicio deben estar clara y visualmente utilizadas, por ejemplo, tarjetas de diferentes colores para representar las clases de servicio o carriles de nado en la pared de tarjetas.
- ❖ Un conjunto de políticas de gestión deben definirse para cada clase de servicio. Sólo las clases de servicio relacionadas con [ítems de mayor riesgo deben incluir actividades de desperdicio tales como estimación.
- ❖ Los miembros del equipo deben estar capacitados para entender las clases de servicio y las políticas relacionadas con ellos.
- ❖ Algunas clases de servicio deben incluir el tiempo de entrega límite.
- ❖ El rendimiento de la fecha límite (porcentaje) debe ser monitorizando para los objetivos de tiempos de entrega.
- ❖ Clases de servicio permiten la auto-organización, facultan a los miembros del equipo y liberan tiempo de administración para concentrarse en el proceso en lugar de en el trabajo.
- ❖ Las clases de servicio cambian la psicología del cliente.
- ❖ Si las clases de servicio se utilizan correctamente y en combinación con una cadencia regular de entrega, muy pocas quejas serán recibidas, aún cuando un monto significativo de ítems no cumpla con el objetivo del tiempo de entrega.
- ❖ La capacidad del sistema kanban debe ser asignada a cada clase de servicio.
- ❖ El porcentaje de capacidad asignado a cada clase de servicio debe estar aliñeadoo con la demanda.

❖ CAPÍTULO 12 ❖

Métricas y reportes de gestión

Aunque la idea es que Kanban sea mínimamente invasivo y cambie la cadena de valor, los roles de trabajo y las responsabilidades lo menos posible, lo que sí cambia es la forma en que el equipo interactúa con sus socios—las partes interesadas externas. Debido a esto, Kanban necesita reportar métricas ligeramente diferentes de las que puede estar acostumbrado ya sea con un enfoque tradicional o un enfoque ágil de gestión de proyectos.

El sistema de flujo continuo de Kanban significa que estamos menos interesados en reportar si un proyecto está “a tiempo” o si un plan específico esta siendo seguido. Lo que es importante es mostrar: que el sistema kanban es previsible y está operando como se diseñó, que la organización exhibe agilidad de negocio, que hay enfoque en el flujo y que hay un claro desarrollo de mejora continua.

En cuanto a previsibilidad, deseamos mostrar que tan bien ejecutamos en base a las promesas de la clase de servicio. ¿Están los ítems de trabajo siendo tratados apropiadamente y, si la clase de servicio tiene un tiempo de entrega límite, que tan bien estamos ejecutando en base a eso? ¿Cuál es el rendimiento de la fecha límite?

Para cada uno de nuestros indicadores, deseamos rastrear la tendencia sobre tiempo para que podamos ver la propagación de variación. Si vamos a demostrar mejora continua, deseamos que la tendencia media mejore sobre el tiempo. Si vamos a demostrar mejoras de previsibilidad, deseamos que la propagación de variación disminuya y que el rendimiento de la fecha límite se incremente.

Rastreando el WIP

Antes de llegar a los indicadores de rendimiento, creo que la métrica más fundamental debe mostrar que el sistema kanban está operando correctamente. Para lograr eso, necesitamos un diagrama de flujo acumulativo que muestra las cantidades de trabajo en progreso en cada estado en el sistema. Si el sistema kanban está fluyendo correctamente, las bandas en la gráfica deben ser suaves y su altura estable.

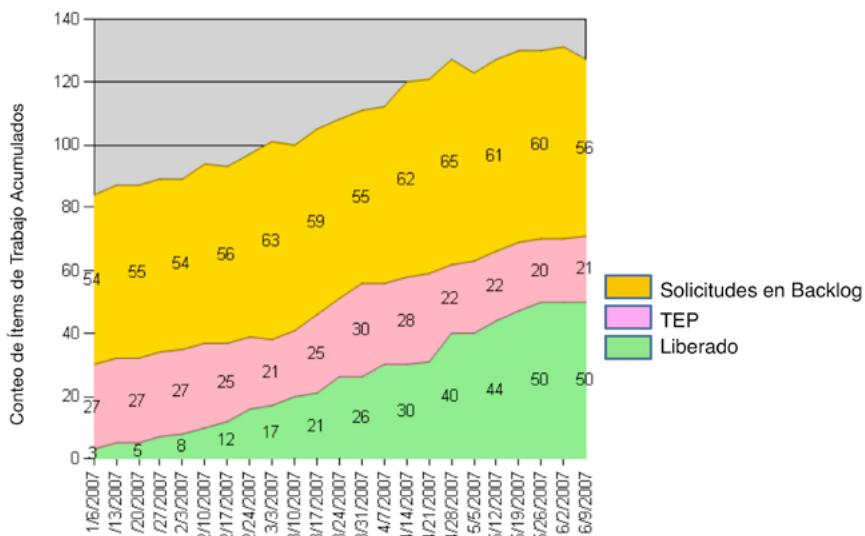


Figura 12.1 Ejemplo del Diagrama de Flujo Acumulativo de un Sistema Kanban

El ejemplo graficado en la Figura 12.1 muestra lo bien que el equipo está manteniendo los límites de WIP. Podemos ver que el WIP (la manda media en color claro) crece a mitad del periodo de tiempo. Al inicio, el límite de WIP es correctamente 27. Al final del periodo, debido a un ajuste de personal, el límite de WIP es correctamente 21. Podemos también leer el tiempo de entrega promedio observando la parte horizontal de la gráfica.

Tiempo de entrega

La siguiente métrica de interés indica con qué previsibilidad nuestra organización entrega contra las promesas en las definiciones de clase de servicio. La métrica subyacente para esto es el tiempo de entrega. Si un ítem es expreso, ¿que tan rápidamente lo llevamos desde la orden hasta producción? Si era de clase estándar, ¿lo entregamos dentro del tiempo de entrega límite? He encontrado que la mejor manera de mostrar estos datos es con un análisis espectral del tiempo de entrega, el cual grafica el tiempo de entrega objetivo contra el acuerdo de nivel de servicio (SLA, en inglés: *Service-Level Agreement—SLA*) para una clase de servicio (ver Figura 12.2).

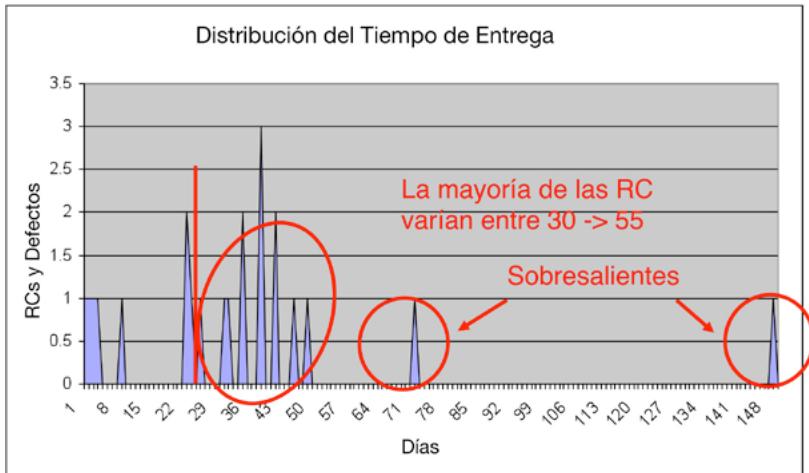


Figura 12.2 Ejemplo del análisis espectral del Tiempo de Entrega

Reportar el tiempo de entrega medio (mean) tiene algunos usos tal como la tarjeta de reporte sobre el rendimiento global (ver Figura 12.3), pero no es muy útil como un indicador de previsibilidad o como un medio para informar sobre oportunidades de mejora.

El análisis espectral es mucho más útil porque nos informa acerca de ítems que apenas fallaron en alcanzar el tiempo objetivo y otros datos estadísticos. En el ejemplo mostrado en la Figura 12.4, tiene sentido investigar las causas raíz del grupo de ítems que fallaron en alcanzar el objetivo. Si estas causas raíz pudieran ser atendidas, el rendimiento de la Fecha Límite (porcentaje de ítems entregados como se esperaba) debería de mejorar.



Figura 12.3 Ejemplo de la tendencia en el Tiempo Medio de Entrega

Porcentajes de Tiempo de Entrega y Fecha de Entrega	Tiempo de Entrega (# promedio de días)			Ejecución de la Fecha de Entrega (%)	
	Objetivo	May-07	Dic 2006 a May 2007	May-07	Dic 2006 a May 2007
Tiempo de Entrega, Ing. Listo para liberar (RCs y soluciones de defectos)	30	32.5	31.1	52	50
Tiempo de Entrega, Ing. Listo para liberar (Solo RCs)	30	32.6	40.4	50	30
Tiempo de Entrega, Ing. Listo para liberar solo defectos)	30	32.5	19.6	55	75

Figura 12.4 Ejemplo de un reporte mostrándole tiempo de entrega medio y el rendimiento de la fecha límite

Rendimiento de la fecha límite

He encontrado útil reportar el Rendimiento de la Fecha Límite para el mes más reciente y para lo que va del año. Usted puede desear también un reporte sobre rendimiento anual (o los últimos 12 meses) para hacer comparaciones. Por ello es útil contar con los últimos 13 meses de datos.

Con los ítems con clase de servicio de Fecha de Entrega Fija, usted puede incluir la métrica del rendimiento de Fecha Límite. En este caso usted estará contestando la pregunta, “¿fue el ítem entregado a tiempo?” Sin embargo, aunque usted tendrá un tiempo de entrega registrado, no es tan interesante como comparar el tiempo de entrega estimado contra el actual. La estimación contra actuales demuestra que tan previsible es el equipo y que tan bien están rindiendo con ítems de servicio de Fecha de Entrega Fija. Recuerde que los ítems de Fecha Fija reciben algún análisis y estimación. El rendimiento de la fecha límite en ítems de fecha fija es un factor que determina la calidad de la estimación inicial. Naturalmente la métrica más importante es si el ítem fue entregado a tiempo antes de la fecha final. La exactitud de la estimación es un indicador de que tan eficientemente el sistema está corriendo. Si se conoce que las estimaciones son inexactas entonces el equipo tenderá a iniciar el trabajo en ítems de fecha fija más temprano para garantizar la entrega. Esto no es óptimo. El rendimiento global sobre ítems de valor y el throughput pueden ser mejorados si la estimación es mejorada.

Throughput

El throughput debe ser reportado como el número de ítems—o alguna indicación de su valor—que fueron entregados en un periodo de tiempo dado, tal como un mes. El throughput debe ser reportado como una tendencia sobre tiempo, como se muestra en la Figura 12.5. La meta es incrementarla continuamente. El throughput es muy similar a la métrica de “velocidad” ágil. Indica cuantas historias de usuario, o puntos de historia, fueron completados en un período dado. Si usted no está utilizando técnicas de requerimientos ágiles sino procesando otras cosas, tales como ítems de especificación funcional, solicitudes de cambio, casos de uso, etc., entonces reporte esos números.

Throughput y Tasa de Producción

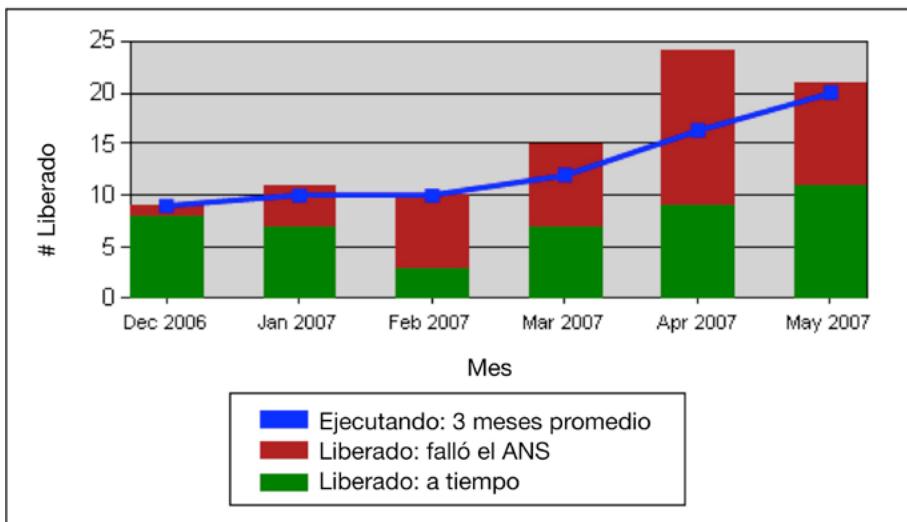


Figura 12.5 Ejemplo de una gráfica de barras mostrando el throughput

En primera instancia es importante poder reportar el número crudo. Conforme su equipo madure y se haga más sofisticado, usted puede reportar el tamaño relativo, tal como el número total de puntos de historia, puntos de función, o alguna otra métrica de calidad. Si su organización es muy sofisticada, usted puede reportar el valor del trabajo entregado en términos de monto de dólares. Al momento de escribir este libro conozco tan solo un equipo, en la BBC de Londres, capaz de reportar el valor en dólares del trabajo entregado.

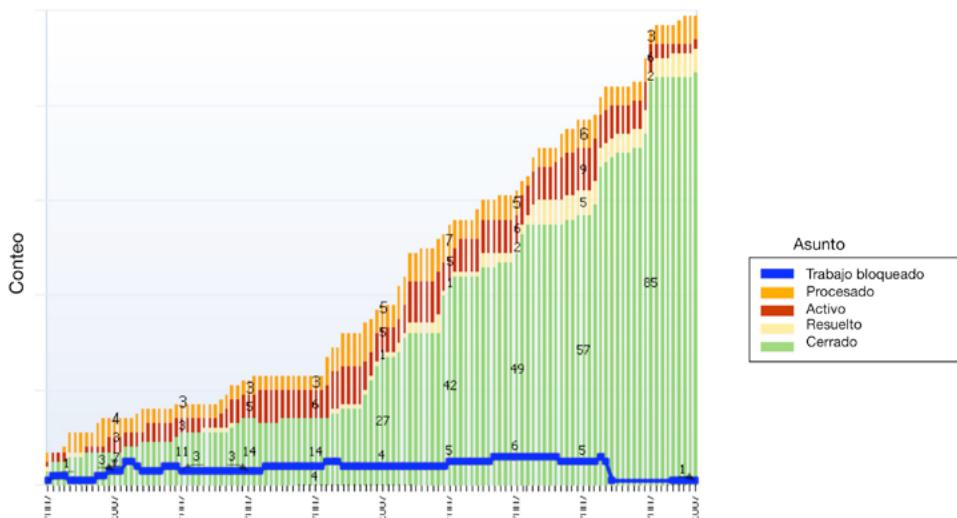
Los datos de throughput son utilizados en Kanban para un propósito enteramente diferente a la velocidad es en un ambiente habitual de desarrollo ágil. El throughput no es utilizado para predecir el monto de entrega en un intervalo de tiempo o cualquier compromiso de entrega específico. El throughput es utilizado como un indicador de que tan bien el sistema (el equipo y la organización) está rindiendo y para demostrar mejora continua. Los compromisos en Kanban están hechos en base al tiempo de entrega y objetivos de fecha de entrega. El Throughput puede ser utilizado en proyectos grandes para indicar el tiempo aproximado de finalización con suficiente buffer para variación.

Asuntos e ítems de trabajo bloqueados

La gráfica de asuntos e ítems de trabajo bloqueados muestra un diagrama de flujo acumulativo de impedimentos reportados superpuestos con una gráfica con el número de ítems de trabajo-en-progreso que se han bloqueado, como se muestra en la Figura 12.6. Esta gráfica nos da una indicación de que tan hábil es la organización para identificar, reportar y gestionar asuntos bloqueados así como su impacto. Si el Rendimiento de Fecha Límite

es pobre, debe haber evidencia correspondiente en esta gráfica demostrando que muchos impedimentos fueron descubiertos y no fueron resueltos lo suficientemente rápido. Esta gráfica puede ser usada diariamente para alertar a la alta dirección de los impedimentos y su impacto. También puede usarse como una tarjeta de reporte a largo plazo para indicar que tan capaz es la organización para resolver impedimentos y para mantener las cosas fluyendo—una medida de capacidad de gestión y resolución de asuntos.

¿Cuantos asuntos ítems de trabajo bloqueados tenemos?



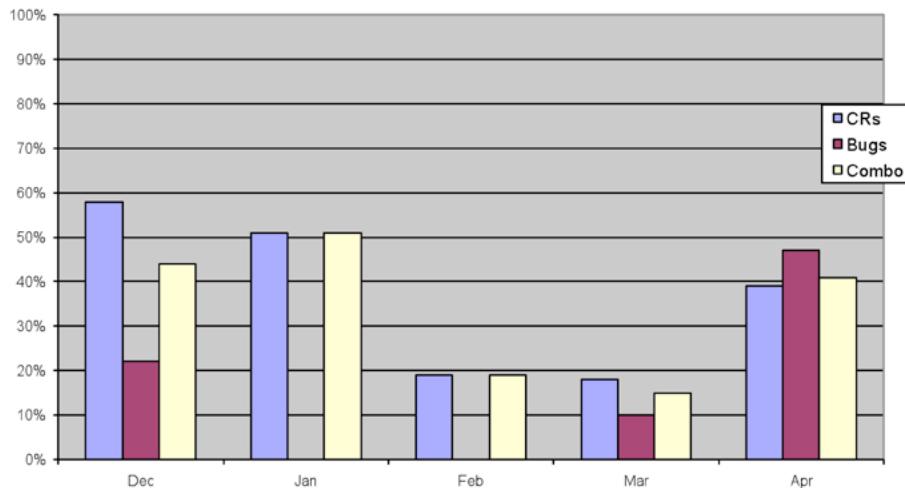


Figura 12.7 Ejemplo de la tasa entre el Tiempo de Entrega y el Tiempo Asignado

La métrica de eficiencia de flujo no es muy útil día a día, pero, de nuevo, puede ser otro indicador de mejora continua.

Calidad inicial

Los defectos representan el costo de oportunidad y afectan tanto el tiempo de entrega como el throughput del sistema kanban. Tiene sentido reportar el número de defectos escapados como un porcentaje contra el WIP total y el throughput. Conforme pasa el tiempo, deseamos ver la tasa de defectos bajar a casi cero, como se muestra en la Figura 12.8.

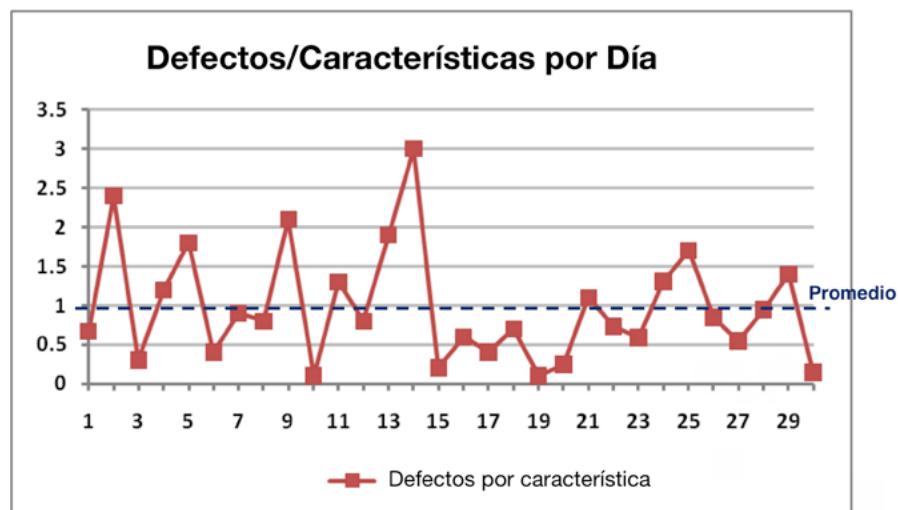


Figura 12.8 Gráfica mostrando los defectos por característica

Carga de falla

La carga de falla rastrea cuantos ítems de trabajo procesamos debido a la pobre calidad anterior—cuantos ítems de trabajo son defectos en producción o nuevas características que han sido solicitadas mediante nuestra organización de servicio a clientes debido a usabilidad pobre o a falta de anticipar la necesidad del cliente apropiadamente. Idealmente, la carga de falla debe bajar con el tiempo. Este es un buen indicador de que estamos mejorando como organización entera y pensando a nivel de sistema.

Para llevar

- ❖ Rastree el WIP con un diagrama de flujo acumulativo para monitorear los límites diarios de WIP.
- ❖ Rastree el tiempo de entrega para cada ítem procesado y reporte tanto la media como el análisis espectral para cada clase de servicio.
- ❖ El tiempo de entrega es un indicador de agilidad del negocio.
- ❖ Rastree el tiempo de entrega estimado contra el actual para ítems de clase de servicio de Fecha de Entrega Fija.
- ❖ Reporte el Rendimiento de Fecha Límite como un indicador de previsibilidad.
- ❖ Los impedimentos bloquean el flujo e impactan el rendimiento tanto de la fecha de entrega como de la fecha límite; reporte los asuntos bloqueados y el número de ítems de trabajo bloqueados en un diagrama de flujo acumulativo con una gráfica sobrepuerta de ítems bloqueados. Úsela para indicar la capacidad para reportar problemas y resolverlos rápidamente.
- ❖ La eficiencia de flujo es la proporción entre tiempo de entrega y tiempo de ingeniería asignada. Indica que tan eficiente la organización es para procesar nuevo trabajo y es un indicador secundario de la agilidad del negocio. También indica cuanto espacio para mejora esta disponible sin cambiar los métodos de ingeniería.
- ❖ La calidad inicial reporta el número de defectos descubiertos por los ingenieros de pruebas dentro del sistema e indica cuanta capacidad es desperdiciada a través de la pobre calidad inicial.
- ❖ La carga de falla reporta el porcentaje de trabajo que es generado a través de alguna falla del sistema y muestra la capacidad remanente en el tablero que podría ser utilizada para características nuevas que agregan valor.

❖ CAPÍTULO 13 ❖

Escalando Kanban con sistemas de dos niveles

Hasta ahora los ejemplos y las historias de las implementaciones de Kanban presentadas aquí se han enfocado en el mantenimiento de software—pequeños cambios en un sistema realizados con entregas rápidas y frecuentes a producción. Hay muchos sistemas que reciben mantenimiento y una porción significativa de los lectores que participan en el desarrollo de software encontrarán los consejos y guías útiles. Del mismo modo, hay muchos más profesionistas de TI involucrados en operaciones y soporte en los que sistemas de ticketing para trabajos rápidos y cortos también son comunes y un enfoque de Kanban sería igualmente útil para ellos. Sin embargo, hay otros profesionistas para quienes el desarrollo de proyectos de tamaño significativo es la norma. Si usted está leyendo esto preguntándose por qué y cómo podría utilizar Kanban en proyectos más grandes y en una cartera de proyectos, espero que el capítulo 5 lo haya persuadido de que Kanban permite cambios culturales importantes y positivos. Los beneficios observados con Kanban son lo suficientemente deseables que nos retan a hacer la pregunta, “¿Cómo haríamos Kanban en grandes proyectos?

Proyectos grandes presentan desafíos considerables. Muchos de los requisitos tienen que ser lanzados juntos. Pasará un número considerable de meses antes de que se haga la primera entrega. El tamaño del equipo es más grande. Puede haber mucho trabajo efectuándose en paralelo. Piezas significativas del trabajo pueden necesitar ser integradas. No todo este trabajo consistirá en desarrollo de software. Por ejemplo, el diseño

del paquete y la documentación pueden necesitar ser integrados con el build del software final antes de que se pueda hacer la entrega.

Entonces, ¿cómo lidiamos con estos retos?

La respuesta está en ver los primeros principios. Los primeros principios de Kanban son limitar el trabajo-en-progreso y arrastrar el trabajo utilizando un sistema de señalización visual. Más allá de ello vemos los principios Lean, los principios Ágiles y el flujo de trabajo y el proceso que ya tenemos establecidos como nuestro punto de partida. Es decir, deseamos limitar el WIP, utilizar controles visuales y señalización y tomar trabajo solamente cuando hay capacidad para hacerlo; pero también deseamos transferencias de lotes pequeños para priorizar por valor, administrar riesgo, hacer progreso con información imperfecta, generar una cultura de alta confianza y responder rápida y graciosamente a los cambios que llegan durante el proyecto.

Con un proyecto grande, como con iniciativas de mantenimiento, usted necesitará acordar la cadencia de priorización para aprovisionar la cola de entrada. La regla general es que una cadencia más alta con reuniones más frecuentes es mejor. Mire los principios de nuevo. ¿Cuáles son los costos de transacción y coordinación de estar sentado con el equipo de marketing o los propietarios de negocio y acordar los siguientes ítems a poner en cola para desarrollarse? Usted tendrá varios puntos de integración o sincronización construyéndose hacia una entrega en lugar de un solo punto de entrega en el otro lado de la cadena de valor. Observe los costos de transacción y coordinación de cualquier integración o sincronización y acuerde la cadencia de nuevo, pero a partir de los primeros principios. Y de nuevo, entre más frecuentemente mejor. Haga la pregunta, “¿Qué involucra reunirse con el negocio para mostrar el trabajo reciente y entonces integrarlo, de tal forma que esté listo para entrega?”

Después de eso, usted deseará acordar los límites de WIP; los principios para pensar a este respecto no cambian. La clase de servicio seguirá haciendo sentido y le ayudará a lidiar adecuadamente con cambios durante el proyecto.

Requerimientos jerárquicos

Usted también necesitará definir los tipos de ítems de trabajo para su proyecto. Muchos proyectos grandes tienen requerimientos jerárquicos. Es común que éstos sean de hasta tres niveles de profundidad. Puede haber también diferentes tipos de requerimientos, tales como requerimientos del cliente que llegan de los propietarios de negocio y requerimientos de producto que llegan de un equipo técnico, de calidad o de arquitectura. Los requerimientos pueden estar descompuestos aún más en requerimientos funcionales, no funcionales, o de calidad de servicio. Aún con desarrollo de software ágil, el cliente puede especificar requerimientos en términos de historias del tamaño de épicas que son descompuestas en historias de usuario que son a su vez descompuestas a un nivel bajo de tareas o unidades pequeñas referidas como “granos de arena”. También he visto épicas descompuestas en

historias arquitectónicas que a su vez son descompuestas en historias de usuario. El desarrollo basado en características también tiene tres niveles de requerimientos—características, conjuntos de características (o actividades) y áreas temáticas.

Ha tenido sentido para los equipos que están adaptando Kanban en proyectos grandes establecer distintos tipos de ítems de trabajo para diferentes niveles de la jerarquía. Por ejemplo, historias épicas son un tipo de ítem de trabajo, e historias de usuario pequeñas son otro. En un proyecto más tradicional, los requerimientos del cliente son de un tipo, mientras que los requerimientos del producto son de otro y los ítems de especificación funcional son un tercer tipo más pequeño.

Normalmente, los equipos escogen rastrear los dos niveles superiores en un tablero de Kanban. En lo personal, no he visto un equipo o proyecto en el que intentaron rastrear tres niveles con Kanban. Algunas herramientas electrónicas ahora soportan requerimientos jerárquicos que permiten al usuario protegerse hacia adentro o hacia fuera a diversos niveles, mostrando solo dos niveles en un momento dado.

Usualmente si hay un tercer nivel más bajo, tal como las tareas en un proyecto Ágil, no son rastreados en una pared de tarjetas de proyecto o dentro del sistema kanban a nivel de equipo. Los desarrolladores independientes pueden escoger rastrear tareas; o tal vez equipos de pequeños multifuncionales escojan rastrear sus tareas localmente, fuera del tablero de proyecto grande y fuera de la vista de los directores y socios de la cadena de valor. Esto no es motivado por la necesidad de esconder información. Es simplemente que el nivel más bajo de actividad no es interesante desde la perspectiva de la cadena de valor o el nivel de rendimiento. El nivel más bajo está frecuentemente enfocado a un esfuerzo y actividad, en lugar de a un valor para el cliente y la funcionalidad.

Al escribir este libro surgió una variante de Kanban para uso personal y fue dirigida por Jim Benson y otros. El Kanban personal se usa en casa y en la oficina a nivel individual, o con pequeños grupos de dos o tres que están colaborando activamente en el mismo conjunto de ítems de trabajo. Es imposible saber al tiempo de escribir este libro si el Kanban personal se reasimilará en el amplio cuerpo de conocimiento o si va a emerger como una disciplina propia.

Desacople la entrega de valor de la variabilidad del ítem de trabajo

Lo que surgió de la mayoría de los equipos de Kanban que rastreaban los dos niveles más altos de requerimientos fue la idea de que el nivel más alto de requerimientos, el que tiene los requerimientos de grano más grueso, generalmente estaban describiendo alguna unidad atómica de valor para el mercado o el cliente. Estas historias de usuario épicas o requerimientos del cliente a menudo se escribían a un nivel tal que tenía sentido liberarlas al mercado. Si el producto ya hubiera estado en mantenimiento, estas solicitudes habrían sido procesadas

y liberadas individualmente. La comunidad Kanban en ocasiones se refiere a este nivel de requerimientos como una “característica mínima negociable” (en inglés: *minimal marketable feature—MMF*). Hay alguna confusión debida a que MMF fue definida por Denne y Cleland-Liang en su libro *Software by Numbers* y no nos adherimos estrictamente a su definición. Preferiría una definición de entrega mínima negociable (en inglés, *minimum marketable release MMR*) que describe un conjunto de características que son coherentes como un conjunto para el cliente y lo suficientemente útiles para justificar los costos de entrega.

No tiene sentido tratar un MMR como un ítem singular que fluye a través de nuestro sistema kanban. Un MMR está hecho de muchos ítems de trabajo. MMR tiene sentido desde una perspectiva de costo de transición de entrega, no desde una perspectiva de flujo. En algunas ocasiones una nueva característica diferenciada, pequeña pero altamente valiosa, puede tener sentido económico para liberarse. Por otro lado, como muchos se han dado cuenta, “el primer MMF siempre es grande” porque la primera entrega de un sistema debe incluir todas las capacidades esenciales para penetrar un mercado y toda la infraestructura para soportarlo. Puede haber una diferencia de dos a tres órdenes de magnitud en el tamaño de los MMFs (o los MMRs). Será problemático un tipo de trabajo con instancias que varían en tamaño hasta mil veces.

Los sistemas kanban no aprecian tan amplio tamaño de variación. Requieren de buffers grandes y WIP excesivo para suavizar el flujo; sin ellos sufrirán amplias fluctuaciones de tiempo de entrega. Buffers largos y mayor WIP significan tiempos de entrega largos y pérdida de agilidad de negocio. ¡La alternativa es peor! Si no tenemos un buffer para variabilidad de tamaño, habrá amplias fluctuaciones en el tiempo de entrega. Como resultado, es imposible ofrecer un tiempo de entrega límite bajo un acuerdo de nivel de servicio (en inglés, *service level agreement SLA*) que puede ser alcanzado consistentemente. Consecuentemente, habrá poca previsibilidad y una pérdida de confianza en el sistema. Diseñar un sistema kanban alrededor del concepto de MMF puede llevar a la pérdida de agilidad de negocio, y/o pérdida de previsibilidad, pérdida de confianza entre TI y el negocio, e insatisfacción general sobre Kanban como un enfoque.

Sin embargo, usando un MMR para activar una entrega en conjunto con un tipo de ítem de trabajo pequeño, de grano fino, es posible minimizar costos y maximizar la satisfacción sobre lo entregado.

Los equipos pueden adaptarse a este reto enfocándose en técnicas de análisis que producen requerimientos de más bajo nivel, tales como historias de usuario o una especificación funcional. Estos serán de grano fino generalmente y tendrán una variación relativamente pequeña en tamaño. Un tamaño ideal sería algo dentro del rango de medio día a cuatro días, más o menos, de esfuerzo de ingeniería.

En un proyecto grande, encontramos que cada ítem de trabajo grande, llamado “Requerimiento”, rastreado con tickets verdes, se descomponía en un promedio de 21 “Características” más pequeñas, rastreadas con tickets amarillos. A pesar de que las características eran escritas de modo que se centraban en el usuario y el valor, eran analizados para ser pequeños y de

tamaño similar. Si éste hubiera sido un proyecto ágil, estos dos niveles habrían sido Épicas, rastreadas en verde y las Historias de Usuario, rastreadas en amarillo.

Los ítems pequeños de grano fino permiten flujo y previsibilidad de throughput y tiempo de entrega, mientras que ítems de grano grueso en el nivel más alto en el tablero nos permiten controlar el número de requerimientos liberables y negociables en progreso en un momento dado.

Adoptando este enfoque de dos niveles desacoplamos la entrega de valor de la variabilidad del tamaño y esfuerzo requerido para entregar ese valor.

Tiene sentido establecer límites de WIP en ambos niveles. Con varios proyectos encontramos que tiene sentido asignar equipos pequeños multifuncionales a cada uno de los requerimientos de alto nivel. Esos equipos tomarían todos los ítems pequeños de grano fino de ese requerimiento de más alto nivel y los fluirían a través del tablero sin pasárselo a otros equipos hasta que el requerimiento estuviese completo y listo para integración o entrega. El equipo tomaría entonces otro requerimiento de grano grueso. También habría la oportunidad de reasignar miembros del equipo, agregando o liberando individuos del equipo, dependiendo del tamaño del siguiente ítem tomado.

Paredes de tarjetas de dos niveles

Los primeros equipos que usaron Kanban en proyectos grandes adoptaron una pared de tarjetas de dos niveles, como se muestra en la Figura 13.1.

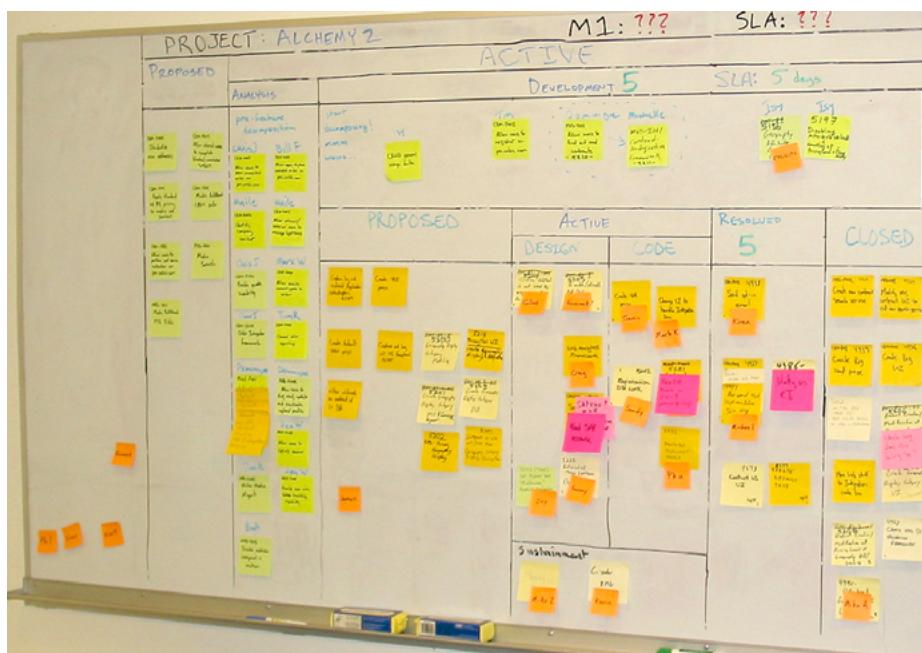


Figura 13.1 Fotografía de un tablero de dos niveles

En esta Figura, los requerimientos de grano grueso se muestran con tickets verdes. Ellos fluyen de izquierda a derecha a través de un conjunto de estados, específicamente, el *backlog*, propuesto (análisis), activo (diseño y desarrollo), resuelto (pruebas) y cerrado.

Los requerimientos que están activos son mostrados a lo largo de la parte superior de la sección media de la fotografía. A su vez, están descompuestos en lotes de características más pequeñas, mostradas con tickets amarillos. Las características fluyen a través de su propio conjunto de estados, específicamente, propuesto (análisis), activo (diseño y código), resuelto (prueba) y hecho. Los estados a través de los cuales las características fluyen son similares a los requerimientos de alto nivel pero no necesitan serlo. Usted puede escoger modelarlo de la manera que considere adecuada. Mi consejo es modelar lo que actualmente hace. No cambie el proceso si lo puede evitar.

Los tickets amarillos están ligados a sus padres etiquetándolos con el número de identificación de sus padres.

Como ejemplo de ello, es posible limitar el WIP en ambos niveles de la jerarquía, pero los tickets amarillos están agrupados en un mismo espacio. No tengo suficiente evidencia del campo para saber si esta es una buena estrategia o no. Lo que si se es que no encajó en este equipo.

Introduciendo carriles de nado

Resulta que hacer coincidir los tickets amarillos de grano fino con sus padres de grano grueso es importante. También parece tener sentido limitar el WIP al nivel bajo, dentro de un equipo multifuncional. Para facilitar este enfoque, algunos equipos innovaron el sistema de pared de tarjetas para introducir carriles de nado horizontales.

En la Figura 13.2 los requerimientos de alto nivel, mostrados en tickets verdes, fluyen a través del mismo conjunto de estados, específicamente, *backlog*, propuesto, activo, resuelto y cerrado. Sin embargo, la sección media ha sido retrasada comparado con la Figura 13.1. Los requerimientos activos de grano grueso en verde están apilados verticalmente en el centro izquierdo. A partir de cada uno de esos tickets verdes se extiende un carril de nado dividido en el mismo conjunto de estados para las características en amarillo de grano fino. El número de carriles de nado es ahora el límite de WIP para los requerimientos negociables por el cliente de grano grueso, mientras que el límite de WIP para las características de grano fino puede ser establecido para cada carril de nado si los equipos individuales deciden hacerlo así. La columna inmediatamente a la derecha de la pila de requerimientos verdes contiene los nombres de los miembros del equipo asignados personalmente. Los tickets anaranjados pequeños adheridos a los tickets amarillos contienen los nombres de los recursos especialistas flotantes, tales como diseñadores de experiencia del usuario y arquitectos de base de datos.

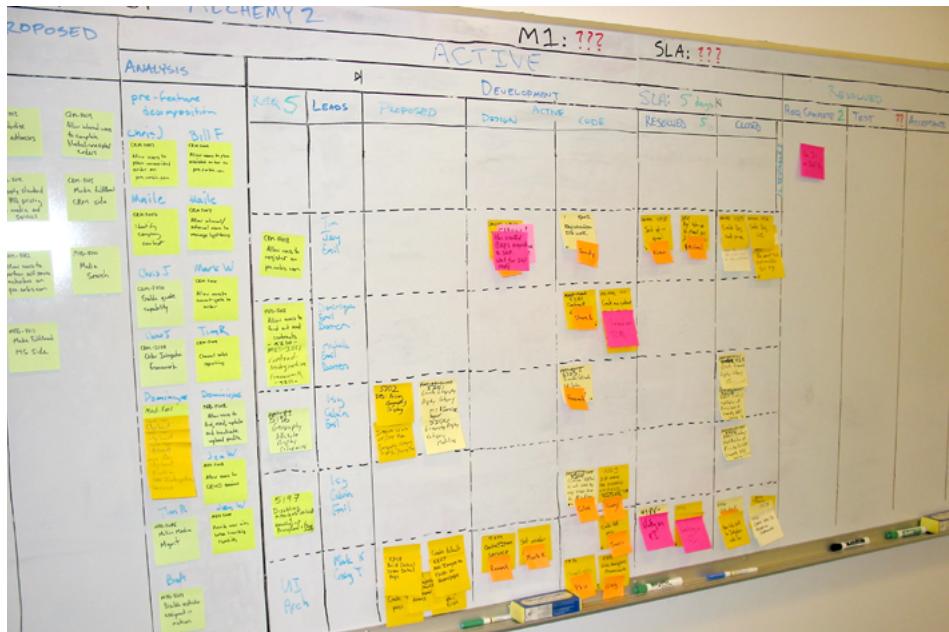


Figura 13.2 Ejemplo de un tablero de dos niveles con carriles de nado

Esta variación de carril de nado de la pared de tarjetas significa que ahora estamos gestionando el WIP de negociación del cliente verticalmente, mientras que estamos gestionando el WIP de las características de baja variabilidad horizontalmente. Este formato demostró ser muy popular y se ha vuelto habitual.

Enfoque alternativo al tamaño de la variabilidad

Otro enfoque para tratar la variabilidad en términos de tamaño es crear diferentes tipos de ítems de trabajo para ítems de distinto tamaño. Los carriles de nado entonces pueden ser asignados para ítems de cada tamaño/tipo. Los límites de WIP serán asignados para cada columna en cada carril, i.e., cada caja en la pared de tarjetas. Debido a que hay baja variabilidad a través de cada carril (ya que los ítems son de tamaño similar), cada carril debe fluir de manera relativamente suave. Esta es una manera de tratar variabilidad sin recurrir a un sistema de dos niveles.

Incorporando clases de servicio

Los dos métodos más obvios para diferenciar tarjetas visualmente en una pared de tarjetas es utilizando tanto color como carriles de nado. Sin embargo, en proyectos grandes todo

ticket tiene habitualmente tres atributos que necesitamos comunicar: el tipo de ítem de trabajo, el nivel en la jerarquía y la clase de servicio. Vale la pena notar que en el ejemplo (ver Figura 13.2) la elección de tener diferentes tipos de ítem de trabajo para diferentes niveles en la jerarquía y entonces usar tanto el color como los carriles de nado para designar el nivel en la jerarquía significa que la jerarquía está siendo sobrecargada con dos métodos de visualización.

Si usted necesita comunicar la clase de servicio en adición al tipo y nivel en la jerarquía de requerimientos, tendrá sentido usar colores para la clase de servicio. Si los tipos son utilizados para propósitos que no son el nivel de jerarquía, por ejemplo, para mostrar bugs o defectos, o adición de valor contra la carga de fallas, usted podría desear escoger otro enfoque. Tal vez introducir un ícono o calcomanía pegada a la tarjeta para significar el tipo; o puede preferir utilizar el color para el tipo y el ícono o calcomanía para la clase de servicio, e.g., una estrella plateada para una solicitud expreso.

Lo que puede ser más fácil, como lo vimos evolucionar en Corbis, es utilizar color para múltiples propósitos, tales como el nivel de jerarquía, el tipo y la clase de servicio. Este enfoque, en el que el color no designa claramente un atributo, parece ser aceptable para los usuarios del sistema kanban y es muy eficiente en términos de opciones disponibles para visualización.

Integración de sistemas

En algunos proyectos grandes, usted puede tener múltiples equipos trabajando en diferentes componentes de un sistema que necesitan ser integrados más adelante. Algunos de estos componentes pueden involucrar hardware o firmware y pueden no ser susceptibles a técnicas modernas de integración continua. Cuando tiene tales componentes que necesitan ser integrados, usted necesita determinar un punto de integración basado en una actividad de planificación de grano grueso de alto nivel. Este punto debe ser tratado como un costo fijo para la entrega de estos componentes dependientes. Esto le permite a cada equipo avanzar independientemente con su sistema kanban y también coordinar la entrega de ítems dependientes cuando son necesitados. La entrega tardía de un ítem dependiente ocasiona retraso excesivo a través del proyecto completo. Este alto costo de retraso justifica tratarlo como un ítem de clase de servicio de fecha fija.

Gestión de recursos compartidos

En proyectos grandes y a través de carteras de proyectos es común compartir algunos recursos especiales; por ejemplo, arquitectura de software, arquitectura y administración de base de datos, prueba a nivel de experiencia del usuario, diseño de la experiencia del

usuario y auditoría de la seguridad del software. Hay tres métodos establecidos en Kanban para tratar estos recursos compartidos.

En el primer método algunos de los ítems de trabajo tienen tickets anaranjados pequeños adheridos a ellos. Estos tickets pequeños muestran el nombre del recurso compartido requerido, digamos Sandy, la arquitecta de la empresa. Al nivel más bajo de intrusión, este acto simple de visualizar el trabajo del recurso compartido es a veces suficiente para coordinar la carga de trabajo de ese individuo. Si varios tickets comienzan a aparecer con el mismo nombre, puede traer preguntas sobre cómo ésta esa persona se está administrando para trabajar en múltiples cosas al mismo tiempo. Esto puede ser suficiente para iniciar una discusión de cambio de política—¿es necesario que todo el trabajo sea visto por esa persona?—o escalar el asunto al siguiente nivel.

El siguiente nivel es reconocer que los recursos compartidos no están disponibles instantáneamente y visualizarlo marcando los ítems que requieran de la atención de un recurso compartido (un ticket anaranjado), como bloqueados hasta que la persona esté trabajando activamente en ellos. Esto tiene el efecto de invocar la función de resolución y gestión de asuntos para resolver la disponibilidad del recurso compartido. También tiene el efecto de remarcarle a la dirección si la disponibilidad de este recurso es un problema y un cuello de botella o no.

El nivel más alto de administración de un recurso compartido es darle a ese recurso su propio sistema kanban. Por ejemplo, la arquitectura de datos de la empresa puede tener su propio sistema kanban, al igual que la experiencia del usuario, la seguridad del sistema y otros. Cada equipo o recurso analizará independientemente su demanda y establecerá sus tipos de ítems de trabajo basados en la fuente de la solicitud y clases de servicio basadas en la prioridad y la respuesta requerida. La demanda será analizada y las políticas para asignación de recursos serán establecidas.

A este nivel ha emergido una arquitectura orientada a servicio para desarrollar el software mismo. Cada grupo dentro de la empresa ofrece su propio conjunto de servicios que son mostrados como acuerdos a nivel de servicio para distintas clases de servicio y tipos de ítem de trabajo. Los clientes de esos recursos compartidos están encolados y son seleccionados para ser procesados, tal y como se ha descrito en éste libro. Si las solicitudes de un cliente dado no están siendo procesadas lo suficientemente rápido, se puede abrir una discusión sobre si el sistema está diseñado correctamente o no y si las políticas de asignación de capacidad y clase de servicio necesitan cambiar o no. Podría proveer evidencia suficiente para realinear o incrementar el personal.

Para llevar

- ❖ Los proyectos grandes deben seguir los principios centrales de Kanban.
- ❖ Los límites de WIP, la cadencia de priorización, la cadencia de entrega y las clases de servicio son técnicas válidas para proyectos grandes.
- ❖ Los proyectos grandes tienden a tener requerimientos jerárquicos; estos niveles de jerarquía deben ser modelados con tipos de ítem de trabajo.
- ❖ Normalmente , los equipos rastrean los dos niveles más altos de requerimientos de la jerarquía en la pared de tarjetas y limitan el WIP a uno o dos niveles.
- ❖ El nivel más alto de requerimientos habitualmente modela los requerimientos negociables por el cliente que crean unidades atómicas que potencialmente podrían ser liberadas individualmente.
- ❖ El segundo nivel de requerimientos se escribe habitualmente en lenguaje centrado al cliente o el usuario y es analizado de tal manera que hace los requerimientos tanto de grano suave como de tamaño similar.
- ❖ El segundo nivel de requerimientos de grano fino facilita el flujo reduciendo variabilidad en el sistema de arrastre kanban.
- ❖ Una pared de tarjetas de dos niveles requerirá visualizar ambos niveles de requerimientos que están siendo rastreados.
- ❖ Los carriles de nado se han convertido en una técnica popular para mostrar la jerarquía y facilitar la limitación del WIP.
- ❖ El WIP de grano grueso está limitado por el número de carriles de nado.
- ❖ El WIP de grano fino puede ser limitado en cada carril de nado, si así se desea.
- ❖ Normalmente , equipos multi-funcionales pequeños están asignados a cada carril de nado.
- ❖ La demanda de recursos compartidos puede visualizarse con calcomanías pequeñas adheridas a ítems de trabajo normales.
- ❖ La disponibilidad no instantánea de recursos compartidos puede ser remarcada con tickets de bloqueo (rosa, en nuestro ejemplo) adicionados al ticket del ítem de trabajo original.
- ❖ Los recursos compartidos deben desarrollar sus propios sistemas kanban.
- ❖ Una red de sistemas kanban para recursos compartidos a través de una cartera de proyectos puede ser concebida como una arquitectura orientada a servicio para el desarrollo de software.

❖ CAPÍTULO 14 ❖

Revisión de operaciones

La reunión previa

Son las 7:30 AM del segundo viernes del marzo de 2007. Estoy temprano en el trabajo porque esta mañana es la cuarta revisión mensual de operaciones de nuestro departamento. Rick Garber, el director de nuestro grupo de ingeniería de procesos de software, está conmigo. Rick tiene el trabajo de coordinar la reunión y la agenda de la revisión de operaciones. Está ocupado imprimiendo el folleto que contiene aproximadamente 70 diapositivas de PowerPoint para la reunión de hoy. Una vez que la impresión ha hecho, nos dirigimos al Harbor Club en el centro de Seattle con una caja de 100 folletos. La revisión de operaciones está programada para comenzar a las 8:30 AM pero un desayuno de buffet caliente es servido a partir de las 8:00 AM. La invitación incluye a todos aquellos de mi organización y de la organización de mi colega Erik Arnold. Con algunas personas en la India, algunas en otras partes de los E.U. y siempre unos pocos que no pueden asistir por razones personales, generalmente tenemos alrededor de 80 gentes atendiendo la reunión.

La invitación incluye a mi jefe, el CIO de Corbis y un número de otros altos directores, nuestros socios de cadena de valor, El grupo externo que atiende la reunión en mayor número es el equipo de Redes y Operaciones de Sistemas encabezado por mi colega Peter Tutak. Después de todo, ellos tienen que recuperar sistemas que fallan en producción, por lo que sienten más las consecuencias de nuestras fallas. Ellos también sienten

el mayor impacto cuando efectuamos nuevas entregas a producción. Por lo que, argumentativamente, ellos son quienes ganan más al participar activamente.

El grupo comienza a llegar a tiempo para disfrutar su desayuno. La sala está en el piso superior de una torre en Seattle que nos brinda hermosas vistas de la ciudad, el puerto, los muelles y la Bahía Elliot. La sala fue configurada con mesas circulares para entre seis y ocho personas en cada una de ellas. Tenemos una pantalla de proyección y un atrio en un lado. Rick gestiona los tiempos con precisión. Cada ponente tiene alrededor de ocho minutos para sus cuatro o cinco diapositivas. Hay pocos buffers de tiempo para permitir la variabilidad que viene con preguntas y discusión. Yo inicio rápidamente con algunas notas. Pido a todos que den marcha atrás en el tiempo hasta finales de enero y recuerden lo que estábamos haciendo en aquel entonces. Me permito recordarles a todos que estamos aquí para revisar el desempeño de la organización para el mes de febrero. Rick ha escogido una buena imagen de los archivos de la compañía para simbolizar un tema del mes y para ayudar a sacudir la memoria, recordando a todos de una actividad clave del mes.

Establezca un tono de negocio desde el principio

Le paso el procedimiento a Rick, quien resume los puntos de acción de gestión del mes pasado e informa sobre su estado. A continuación presentamos a nuestro analista de finanzas, quien ofrece un resumen de los resultados de la empresa durante el mes—la causa del retraso hasta el segundo viernes del mes siguiente es para que podamos tener los datos financieros después de que los libros del mes anterior se han cerrado. Ella da un sumario de los detalles del presupuesto para los centros de costo tanto mío como de Erik. Vemos lo programado versus lo real para todas las áreas presupuestarias principales, así como las metas de número de personal. Discutimos las requisiciones abiertas y alentamos a los miembros del equipo a presentar candidatos para las posiciones abiertas. Al salir de este primer segmento, todos los asistentes saben lo bien que la empresa se está desempeñando y lo bien que el grupo de ingeniería de software es gestionando con relación al presupuesto y por lo tanto, la cantidad de holgura que tenemos para comprar artículos como los monitores de pantalla plana y equipos nuevos. El propósito de abrir con los números financieros es recordar a todos en el equipo que estamos en un negocio, no estamos simplemente llegando a un lugar todos los días para divertirnos con unos y ceros con un grupo de amigos.

Tener invitados amplía la audiencia y agrega valor

El siguiente orador es un invitado—un vicepresidente de otra parte de la empresa. Tuve la brillante idea de que si queríamos que nuestros socios de la cadena de valor se interesaran, deberíamos mostrar interés en ellos e invitarlos a formar parte de la presentación. Le ofrecimos a nuestro invitado 15 minutos y él los tomó. Tuvimos una presentación sobre

las operaciones de ventas, la parte de la empresa que cumple las órdenes del cliente y garantiza la entrega del producto. Aunque algunos de los negocios de Corbis se hacen en la web y se entregan vía electrónica, no todo lo que la empresa ofrece es entregado como una descarga; un departamento completo cumple órdenes más complejas para las agencias de publicidad y empresas de medios de comunicación. Mi colega, Erik Arnold, tuvo la brillante idea de pedirle al cliente que patrocinara nuestro desayuno para mantener nuestros costos bajo control. Eso también funcionó. Durante los próximos meses nuestro equipo aprendió acerca de muchos aspectos de la empresa y los altos dirigentes de toda la compañía se enteraron de lo que hacíamos, como lo hacíamos y lo fuertemente que estábamos tratando de hacer frente a nuestros problemas. Nueve meses más tarde, los ejecutivos hablaban abiertamente sobre lo bien que el equipo de TI estaba dirigido y como su unidad de negocios debería estar siguiendo nuestro ejemplo.

Agenda principal

Una vez que nuestro orador invitado terminó, pasamos a la parte principal de la reunión. Cada director tuvo ocho minutos para hacer una presentación sobre rendimiento de su departamento. Continuamos la reunión con actualizaciones sobre proyectos específicos de nuestra oficina de gestión de programas. Cada uno de los directores inmediatos de equipo se levantó y pasó cinco minutos presentando rápidamente sus métricas. En general siguieron el formato desplegado en el capítulo 12: presentaron información sobre tasas de defectos, tiempo de entrega, throughput, eficiencia del valor agregado y, ocasionalmente, un reporte específico que penetraría algún aspecto de nuestro proceso sobre el cual necesitaban más información. Atendían preguntas, comentarios y sugerencias por varios minutos.

Este cuarto mes de la Revisión de Operaciones, marzo de 2007, fue particularmente interesante. La primera Revisión de Operaciones fue en diciembre. Todos atendieron la reunión, casi el 100 por ciento. Había mucha curiosidad y muchos comentarios posteriores tales como, “Nunca había visto este nivel de transparencia en todami carrera” y “Eso estuvo muy interesante”. La retroalimentación más útil fue, “¿podemos tener un buffet caliente en lugar de frío la próxima vez?” Por lo que agregamos desayuno caliente. El segundo mes la gente dijo, “Si, otro buen mes. ¡Lo cual es interesante! ¡Gracias por el desayuno caliente!” Al tercer mes algunos de los desarrolladores preguntaron, “¿porqué me tengo que levantar tan temprano?” y “¿es esto un buen uso de mi tiempo?”

Lo que sucedió en el cuarto mes es que revisamos un problema significativo. La empresa había adquirido un negocio en Australia. Se le pidió a TI que apagara todos los sistemas de IT de la subsidiaria australiana y migrara todos los 50 usuarios a los sistemas de Corbis. La solicitud tenía una fecha arbitraria y urgente. La fecha estaba basada en un ahorro de costo del estilo de la “economía de escala” que había justificado parcialmente el precio de adquisición, por lo que había costo de retraso involucrado. La solicitud llegó como un ítem solo en

nuestra cola de mantenimiento. Era lo suficientemente grande como para haber justificado 10 tickets, pero lo tratamos como uno solo. El efecto de un ítem demasiado grande como éste entrando al sistema kanban está bien entendido en la ingeniería industrial. Obstruye el sistema y extiende el tiempo de entrega para todo lo que llega detrás de ella. Y eso nos sucedió a nosotros. El tiempo de entrega brincó, en promedio, de 30 a 55 días. La teoría de colas también nos dice que reducir el *backlog* estando totalmente cargado toma mucho tiempo. Descubrimos que nos llevaría cinco meses el recuperar el tiempo de entrega objetivo.

Adicionalmente, teníamos una entrega que había requerido una solución de emergencia.

Repentinamente la habitación se iluminó con preguntas, comentarios y debate. Después de tres meses de datos aburridos y buenos teníamos una historia que contar. El personal estaba sorprendido de que nosotros (la dirección) estudiésemos dispuestos a hablar abiertamente sobre los problemas y qué hacer con ellos y que la revisión de operaciones no fuese tan sólo para presumir de lo buenos que somos y tan sólo presentar los datos buenos. Nadie del personal cuestionó de nuevo por qué llevamos a cabo la reunión de cada mes.

La reunión terminó con Rick abreviando los ítems de acción para la dirección de las discusiones de la mañana y agradeciendo a todos por asistir. Eran las 10:30 AM y el momento de atravesar la calle para regresar a la oficina.

La clave de la transición lean

Hay muchas de cosas importantes que entender sobre Revisión de Operaciones. En primer lugar, creo que Revisión de Operaciones es la piedra angular de la transición a Lean y de la implementación del método Kanban. Es una retrospectiva objetiva, basada en datos, sobre el desempeño de la organización. Va más allá de cualquier proyecto y se establece una expectativa de gestión cuantitativa objetivo, basada en datos, en lugar de la gestión cualitativa, subjetiva y anecdótica, que es la práctica más establecida con proyectos ágiles y retrospectivas de iteración. La revisión de operaciones proporciona el círculo de retroalimentación que permite el crecimiento de la madurez de la organización y mejora continua a nivel de la organización. Sinceramente, creo que es esencial para lograr una transición exitosa de Lean (o Ágil) a nivel de empresa.

La cadencia apropiada

También pienso que la revisión de operaciones tiene que ser mensual. Más a menudo puede ser una carga para la toma de datos y el tiempo necesario para la reunión significa que se desea no hacerlo demasiado a menudo. Ajustar tal reunión en dos horas es un reto. Si no fuera una reunión basada en datos, llena de gráficas y reportes, no sería posible. Una reunión subjetiva y de estilo anecdótico en esa escala no se podría completar en dos horas. Una retrospectiva típica de proyecto tarda más de dos horas, así que imagínese tratando de

realizar una retrospectiva de toda la organización y completarla en dos horas utilizando un estilo de análisis de ventajas y desventajas. Parte del secreto para mantener la duración de la reunión breve es llevar a cabo en base a datos objetivos; mantenga la agenda apretada y adminístrela durante toda la reunión.

Puede haber una tendencia a desechar llevar las revisiones de operación menos frecuentemente; trimestralmente es común. Mi experiencia con revisiones de operaciones trimestrales fue durante mi tiempo en la división PCS de Motorola. Mi observación sobre esas reuniones es que eran sesiones de reporte y gestión de la alta dirección y no sesiones organizacionales diseñadas para gestionar la mejora continua y la madurez de la organización. Un período trimestral es demasiado infrecuente como para realmente gestionar un programa de mejora. Los datos tienen a menudo cuatro meses de antigüedad para cuando llega a la revisión trimestral. Un trimestre es un espacio de tiempo demasiado largo para revisarlo en una sola reunión, por lo que las revisiones tienden a ser superficiales. Los reportes y métricas tienden a ser indicadores de retraso y se enfocan en reportar ejecución contra el objetivo de los directivos.

Las reuniones trimestrales parecen atractivas porque se sienten más eficientes—una reunión de dos horas cada trimestre en lugar de cada mes. También cuestan menos en una base anual—tan solo cuatro reuniones en lugar de 12. Después de dejar Corbis al inicio de 2008, mi ex-jefe redujo la cadencia de las revisiones de operaciones a trimestrales para ahorrar dinero. Después de tres trimestres y con ese jefe también habiendo dejado la empresa, el nuevo liderazgo cuestionó el valor de las reuniones y decidió cancelarlas por completo. Después de varios meses el rendimiento de la organización supuestamente se había depreciado considerablemente y el nivel de madurez organizacional había caído de nuevo, según reportes, caído de nuevo a aproximadamente del equivalente del nivel 4 de CMMI a nivel 2 de CMMI; de administrado cuantitativamente a apenas administrado.

Podemos sacar varias cosas de esto. La pérdida de un círculo de retroalimentación redujo las oportunidades de reflexión y adaptación que podían llevar a mejoras. Eliminar una reunión basada en revisión de ejecución objetiva de la organización envió un mensaje de que el liderazgo ya no se preocupaba por el rendimiento. El resultado fue un paso significativo hacia atrás en la madurez organizacional y el rendimiento en términos de previsibilidad, calidad, tiempos de entrega y throughput.

Demostrando el valor de los directores

La revisión de operaciones también le mostró al personal lo que los directores hacen y como la gestión agrega valor a sus vidas. También ayuda a capacitar la fuerza de trabajo para que piensen como directores y entiendan cuándo hacer intervenciones y cuándo dar un paso atrás para dejar que el equipo se auto-organice y resuelva sus propios asuntos. La revisión de operaciones ayuda a desarrollar respeto entre los trabajadores de conocimiento individuales, sus directores y entre diferentes capas de administración. Incrementando el respeto se genera confianza, promueve colaboración y desarrolla el capital social de la organización.

El enfoque organizacional fomenta kaizen

Mientras que las retrospectivas de proyecto individuales son siempre útiles, una revisión de operaciones a nivel de la organización fomenta la institucionalización de cambios, mejoras y procesos. Promueve mejoras para difundirse de manera viral a través de la organización y genera una pequeña rivalidad de paredes de tarjetas entre proyectos y equipos que motiva a todos a mejorar su rendimiento. Los equipos desean demostrar cómo pueden ayudar a la organización con mejor previsibilidad, más throughput, tiempos de espera más cortos, costos bajos y mayor calidad.

Un ejemplo anterior

Yo no inventé las revisiones de operación. Son bastante comunes en muchas compañías grandes. Sin embargo, aprendí como hacerlas de esta manera objetiva y a nivel de unidades de negocio cuando trabajé en Sprint PCS en 2001. Mi jefe, el vicepresidente y director general de sprintpcs.com, las instituyó por razones muy similares. El deseaba desarrollar la madurez de su organización—una unidad de negocio de 350 personas responsable del sitio web, todo el comercio electrónico y servicio a clientes en línea para el negocio de telefonía celular de Sprint. En sprintpcs.com llevábamos a cabo la revisión de operaciones cada tercer viernes del mes a las 2:00 PM. Duraba dos horas y reunía alrededor de 70 empleados senior y directores de las unidades de negocio, más invitados a nivel de director o alta dirección de los socios de los niveles anteriores y posteriores del proceso. Líderes senior, incluyendo el Jefe de Mercadotecnia (en inglés: *Chief Marketing Officer*) y el VP de Planificación Estratégica, atendían las reuniones regularmente. El formato era muy similar al de Corbis. Estaba basado enteramente en datos objetivos. Cada director presentaba sus propios datos. La reunión iniciaba con datos financieros. El calendario era planificado y administrado celosamente. Después de la reunión todos podían ir a sus casas temprano. La reunión se llevaba a cabo fuera de las premisas, en el campus de un colegio local. Mientras que sprintpcs.com había tenido dificultades con las técnicas de desarrollo de software ágil, la revisión de operaciones era un elemento clave en el desarrollo de la madurez organizacional y en el mejoramiento del gobierno corporativo. Le mostraba al personal que los directores estaban haciendo una diferencia y sabían cómo administrar y le dio al personal y los directores de línea una oportunidad de mostrarle a los líderes senior como podrían ayudar y dónde necesitaban intervenciones para realmente hacer una diferencia.

Dados estos dos experimentos sobre un periodo de cuatro años a través de la última década me he convencido de que la revisión de operaciones es una pieza crítica en una transición Lean-Ágil exitosa y un componente vital en el desarrollo de la madurez organizacional.

Para llevar

- ❖ Las revisiones de operaciones deben ser a nivel de la organización.
- ❖ Las revisiones de operaciones se enfocan en datos objetivos.
- ❖ Cada departamento debe reportar sus propios datos.
- ❖ Las presentaciones deben ser cortas y normalmente deben reportar métricas e indicadores similares a los discutidos en el capítulo 12.
- ❖ Dirigir con información financiera remarca que la función de ingeniería de software es parte de un negocio más amplio y que buen gobierno es importante.
- ❖ Una cadencia mensual para revisiones de operaciones parecer ser más o menos correcta. De manera más continua es una carga en cuanto a el compromiso de tiempo, colección de datos y preparación. Hacerlo más espaciadamente tiende a reducir el valor y socava la naturaleza de la reunión.
- ❖ Las reuniones deben mantenerse cortas, normalmente de dos horas.
- ❖ Las revisiones de operaciones deben usarse para proveer un círculo de retroalimentación y dirigir mejora continua a nivel de empresa o de unidad de negocio.
- ❖ Las revisiones de operaciones muestran a los contribuyentes individuales cómo la dirección puede agregar valor a sus vidas y lo que los directores efectivos pueden hacer.
- ❖ Las revisiones de operaciones efectivas generan confianza mutua entre los directores y los trabajadores.
- ❖ Las partes interesadas externas que atienden las revisiones de operaciones tienen una oportunidad de ver cómo funcionan los grupos de ingeniería de software y grupos de TI, y para entender sus asuntos y retos. Esto fomenta la confianza y la colaboración.
- ❖ Las revisiones de operaciones deben examinar los problemas con datos malos tanto como estar complacidos por el éxito y halagar las virtudes de los equipos que tienen buenos resultados.
- ❖ Llevar a cabo las reuniones fuera de las oficinas parece ayudar a enfocar la mente de los presentes.
- ❖ Proveer comida parece fomentar la asistencia de la gente.
- ❖ La involucración de líderes senior indica que la organización toma el rendimiento y la mejora continua seriamente.

- ❖ Señalar un interés serio en la ejecución, la mejora continua y la dirección cuantitativa es vital en el desarrollo de una cultura Kaizen entre la fuerza de trabajo general.
- ❖ Las revisiones de operaciones han mostrado que dirigen directamente al incremento de niveles de madurez organizacional.
- ❖ Las sugerencias para la mejora deben ser capturadas como ítems de acción gerencial y su progreso debe ser revisado al inicio de las siguientes y subsiguientes reuniones.
- ❖ Los directores deben ser responsables y deben demostrar que dan seguimiento de las sugerencias.

❖ CAPÍTULO 15 ❖

Comenzando una iniciativa Kanban de cambio

Comenzar con Kanban no es como las iniciativas de un proceso habitual que usted haya tenido en el pasado. Es importante establecer los fundamentos para un éxito a largo plazo. Para lograr eso, es necesario entender las metas detrás de utilizar el enfoque de cambio Kanban. Este libro está subtitulado, “Cambio evolutivo exitoso para su negocio de tecnología”. Hice esto para remarcar el punto de que la razón principal para adoptar Kanban es la gestión de cambio. Todo lo demás es secundario.

Kanban acelera la madurez y capacidad organizacional

La técnica de Kanban está diseñada para minimizar el impacto inicial de los cambios y reducir la resistencia de la adopción de los cambios. Adoptar Kanban debe cambiar la cultura de la organización y ayudarla a madurar. Si se hace correctamente, la organización se transformará en una entidad que adopta cambios rápidamente y se hace buena para implementar cambios y mejoras de proceso. El Software Engineering Institute (SEI) se refiere a esto como una capacidad en Innovación y Despliegue Organizacional (OID)²⁰ dentro de su Modelo de Integración de Capacidad de Madurez (CMMI). Se ha demostrado que organizaciones que alcanzan este alto nivel de capacidad en la gestión de cambios

pueden adoptar métodos ágiles tales como Scrum más rápido y de mejor manera que organizaciones menos maduras²¹.

Cuando implemente Kanban por primera vez buscará optimizar los procesos actuales y cambiar la cultura organizacional en lugar de reemplazar los procesos actuales por otros que puedan proveer mejoras económicas dramáticas. Esto ha llevado a la crítica²² de que Kanban tan solo optimiza algo que necesita cambio. Sin embargo, ahora existe evidencia empírica considerable²³ de que Kanban acelera el logro de altos niveles de madurez y capacidad organizacional en áreas de proceso central de alta madurez tales como Análisis y Resolución Causal (CAR) e Innovación y Despliegue Organizacional. Cuando escoja utilizar Kanban como un método para dirigir cambios en su organización, se está suscribiendo al punto de vista de que es mejor optimizar lo que ya existe porque hacer eso es más fácil, más rápido y con mínima resistencia. Usted debe entender también que los aspectos de Kanban de juego colaborativo contribuirán a un cambio significativo en su cultura corporativa y su madurez. Este cambio cultural permitirá cambios mucho más significativos más adelante, de nuevo, con menor resistencia que si intentara hacer esos cambios inmediatamente. Adoptar Kanban es una inversión a largo plazo en la capacidad, madurez y cultura de su organización. Su intención no es ser una reparación rápida.

Debido a esto, es poco probable que usted pueda dirigir la adopción de Kanban a través de una iniciativa de transición planificada y un programa de entrenamiento prescrito. Es cierto que será necesario algún entrenamiento. Los miembros del equipo y otras partes interesadas deben entender tales bases tales como la relación entre el WIP y el tiempo de entrega y que limitando la cantidad de WIP de manera estricta mejorará la previsibilidad del tiempo de entrega. Puede ser también necesario proveer un resumen de oportunidades de mejora probables tales como cuellos de botella, desperdicio y variabilidad. Conforme estas oportunidades de mejora son descubiertas, puede ser necesario más entrenamiento sobre nuevas habilidades y técnicas. Por ejemplo, si los defectos son una fuente mayor de desperdicio, el equipo de desarrollo puede requerir entrenamiento en técnicas que reducirán defectos grandemente y mejorarán la calidad del código, tales como integración continua, pruebas unitarias y programación en parejas.

Sin embargo, en lugar de perder mucho tiempo al inicio en educación, es más importante que obtenga un consenso alrededor de Kanban y comience a utilizarlo. Este capítulo busca establecer los fundamentos para una transición exitosa a Kanban y le provee con una guía de doce pasos para comenzar.

Aunque nuestra meta principal con Kanban es introducir cambios con mínima resistencia, debe haber otras metas. El cambio por el cambio mismo no tiene sentido. Las otras metas deben reflejar necesidades de negocio legítimas tales como alta calidad y entrega previsible. Las metas listadas aquí tienen la intención de ser ejemplos. Las metas específicas de su organización pueden ser diferentes. El paso 1 de su proceso debe ser acordar las metas por las cuales introducir Kanban en su organización.

La meta principal de nuestro sistema kanban

Estamos haciendo Kanban porque creemos que provee una mejor manera de introducir cambios. Inicialmente Kanban busca cambiar lo menos posible, por lo que la resistencia mínima debe ser nuestra primera meta.

META 1 OPTIMIZAR LOS PROCESOS EXISTENTES

Los procesos existentes deben ser optimizados mediante la introducción de visualización y limitando el trabajo-en-progreso para catalizar los cambios. En cuanto a roles y responsabilidades, no cambian para minimizar la resistencia de los empleados.

Las metas secundarias de nuestro sistema kanban

Sabemos que Kanban nos permite entregar los seis elementos de la Receta para el Éxito (del capítulo tres). Sin embargo, podemos desear modificar ligeramente la redacción de las metas y expandir algunos de los puntos de la receta para reflejar que esos puntos nos pueden ayudar a la consecución de más de una meta.

META 2 ENTREGAR CON ALTA CALIDAD

Kanban nos ayuda a enfocarnos en la calidad limitando el trabajo-en-proceso y permitiéndonos definir políticas alrededor de lo que es aceptable antes de que un ítem de trabajo pueda ser arrastrado al siguiente paso en el proceso. Estas políticas pueden incluir criterios de calidad. Si, por ejemplo, establecemos una política estricta para que las historias del usuario no puedan ser arrastradas a las pruebas de aceptación hasta que se pasen todas las demás pruebas y los defectos sean resueltos. Estamos “deteniendo la línea” efectivamente hasta que la historia esté en una condición suficientemente buena para continuar. Con un equipo nuevo en Kanban puede que no implementemos una regla tan estricta, pero debe haber algunas políticas relacionadas con la calidad que enfocan al equipo a desarrollar código funcional con un bajo número de defectos.

META 3 MEJORAR LA PREVISIBILIDAD DEL TIEMPO DE ENTREGA

Sabemos que el monto de WIP está directamente relacionado con el tiempo de entrega y que también hay una correlación entre el tiempo de entrega y un crecimiento no lineal en la tasa de defectos. Por ello tiene sentido que

deseemos mantener el WIP pequeño. Nuestras vidas serán más fáciles si acordamos limitarlo a una cantidad fija. Esto debe hacer los tiempos de entrega un tanto cuanto confiables y nos ayudará a mantener bajas las tasas de defectos.

META 4 MEJORAR LA SATISFACCIÓN DE LOS EMPLEADOS

Aunque la satisfacción de los empleados a menudo recibe atención, de palabra, en la mayoría de las empresas es raramente una prioridad. Con demasiada frecuencia, los inversionistas y la alta dirección ven a los recursos humanos como fungibles y fácilmente reemplazables. Esto refleja una vía centrada-al-costo en su enfoque de gestión o inversión. No toma en cuenta el impacto tremendo en el rendimiento que viene con una fuerza de trabajo bien motivada y con experiencia. La retención de personal es importante. A medida que la población de desarrolladores de software crece en edad, se preocupan más por el futuro de sus vidas. Muchos lamentan haber desperdiciado la primera etapa de su vida profesional, encerrados en una oficina, esclavizados en un pedazo de código que falló en alcanzar las expectativas del mercado y se hizo obsoleto poco después de ser entregado.

El equilibrio entre el trabajo y la vida no se trata tan solo de equilibrar el número de horas que uno pasa en el trabajo con el número de horas que tienen disponible para su familia, amigos, hobbies, pasiones, anhelos. También tiene que ver con proveer fiabilidad. Digamos, por ejemplo, que un miembro del equipo con pasión por el arte desea tomar una clase de pintura en la escuela secundaria local. Comienza a las 6:30 PM todos los miércoles por diez semanas. ¿Puede su equipo proveerle a ese individuo la certidumbre de que estará libre para dejar la oficina a tiempo cada semana con el fin de atender la clase?

El proveer un buen equilibrio entre el trabajo y la vida hará de su empresa un lugar más atractivo en el mercado local. Ayudará a motivar a los empleados y les dará a los miembros de su equipo la energía para mantener altos niveles de rendimiento por meses o años. Es una falacia que se obtenga un alto rendimiento de los trabajadores de conocimiento cuando se les sobrecarga de trabajo. Puede ser cierto tácticamente por unos pocos días, pero no es sostenible más allá de una o dos semanas. No sobrecargar a sus equipos y brindarles un buen equilibrio entre trabajo y vida es muy buen negocio.

META 5 PROVEER HOLGURA PARA HACER POSIBLE LA MEJORA

Aunque el tercer elemento de la Receta para el Éxito—equilibrar la demanda contra el throughput—puede utilizarse para evitar sobrecargar a los miembros del equipo y permitirles un equilibrio confiable entre trabajo y vida, tiene un segundo efecto. Genera holgura en la cadena de valor. Debe haber un cuello de botella en su organización. Toda cadena de valor tiene uno. El throughput entregado en los niveles posteriores del proceso está limitado al throughput del cuello de botella independientemente de qué tan lejos en los niveles anteriores del proceso pueda estar. Por ende, cuando usted equilibra la demanda de entrada contra el throughput crea tiempo ocioso en todos los puntos de su valor excepto en el recurso con cuello de botella.

La mayoría de los directores son reacios a la idea de tiempo ocioso. Han sido entrenados en general para administrar por utilización (o eficiencia, como se le llama a menudo), e inherentemente sienten que se pueden hacer cambios para reducir costos si existe tiempo ocioso. Esto puede ser cierto, pero es importante apreciar el valor de la holgura.

La holgura puede ser utilizada para mejorar la respuesta a solicitudes urgentes y proveer recursos para permitir la mejora de procesos. Sin holgura, los miembros del equipo no pueden tomar tiempo para reflexionar sobre la manera cómo hacen su trabajo y como lo podrían hacer mejor. Sin holgura no pueden tomar tiempo para aprender nuevas técnicas o mejorar sus herramientas o sus habilidades y capacidades. Sin holgura no hay liquidez en el sistema para poder responder a solicitudes urgentes o cambios tardíos. Sin holgura no se tiene agilidad táctica en el negocio.

META 6 SIMPLIFIQUE LA PRIORIZACIÓN

Una vez que un equipo es capaz de enfocarse en la calidad, limitar el WIP, entregar continuamente y equilibrar la demanda contra el throughput, tendrán una capacidad de desarrollo de software que es confiable: ¡una máquina para hacer software! Una fábrica de software si así lo desea. En cuanto esta capacidad es establecida, le convendrá al negocio hacer uso óptimo de ella. Para lograrlo se necesita un método de priorización que maximice el valor del negocio y que minimice los riesgos y los costos. Idealmente, un esquema de priorización que optimice el rendimiento del negocio (o del departamento de tecnología) es sumamente deseable.

Los campos de ingeniería de software y de gestión de proyectos han estado desarrollando esquemas de priorización desde que los proyectos de software se iniciaron, probablemente hace 50 años. La mayoría de los esquemas eran simples. Por ejemplo, Alto, Medio y Bajo proveen tres clasificaciones simples. Ninguno de ellos tiene significado directo alguno para el negocio. Algunos esquemas más elaborados llegaron a usarse con el advenimiento de las metodologías de desarrollo ágil de software tales como MoSCoW (Debe tener, Debería tener, Podría tener, No tendrá. En inglés: "*Must have*", "*Should have*", "*Could have*", "*Won't have*"). Otros métodos tales como Desarrollo Basado en Características (en inglés, feature-driven development FDD) utilizaron una versión modificada y simplificada de la técnica de Análisis Kano, el cual es popular en las empresas japonesas. Algunos otros promueven un orden estrictamente numérico (1, 2, 3, 4...) para el valor del negocio o el riesgo técnico. El riesgo con este último esquema está en que en ocasiones crea un conflicto entre los ítems de alto riesgo que deben ser priorizados primero y los ítems de alto valor que también deben ser priorizados primero.

Todos estos esquemas sufren de un problema fundamental. A fin de responder a cambios en el mercado y eventos evolutivos es necesario volver a priorizar. Imagine, por ejemplo, que tiene un *backlog* de 400 requerimientos priorizados en un orden estrictamente enumerado –1 a 400 – y está entregando incrementalmente utilizando un método de desarrollo ágil en iteraciones de un mes. Usted tendría que volver a priorizar el *backlog* remanente hasta un máximo de 400 ítems cada mes.

Desde mi experiencia, pedirles a los propietarios del negocio que prioricen las cosas es un reto. La razón es muy simple. Existe mucha incertidumbre en el mercado y en el ambiente de negocio. Es difícil predecir el valor futuro de una cosa contra la otra, o cuando algo va a ser necesario, o si algo más puede ser valioso más rápidamente o no. Pedirle a un propietario de un negocio que priorice el *backlog* de los requerimientos de un sistema de tecnología es hacerle un conjunto de preguntas difíciles para las cuales las respuestas son inciertas. Cuando las personas están inciertas, tienden a reaccionar mal. Puede que se muevan despacio. Puede que se rehúsen a cooperar. Pueden volverse incómodos y disfuncionales. Podrían simplemente reaccionar procrastinando y cambiando de parecer constantemente, modificando planes de proyectos aleatoriamente y desperdiциando mucho tiempo del equipo reaccionando a esos cambios.

Lo que se necesita es un esquema de priorización que retrase los compromisos lo más posible y que haga una pregunta simple que sea fácil de responder. Kanban provee estos solicitándoles a los propietarios de los negocios que aprovisionen los espacios vacíos en la cola mientras que les provee con un tiempo de entrega confiable y una métrica de ejecución de fecha límite.

Ya tenemos seis metas valiosas para nuestro sistema kanban y para muchos negocios, lo cual puede ser suficiente. Sin embargo, yo y otros adoptantes de Kanban hemos descubierto que hay otras dos metas aún más valiosas que son tanto posibles como deseables.

META 7 PROVEER TRANSPARENCIA EN EL DISEÑO Y OPERACIÓN DEL SISTEMA

Cuando comencé a utilizar sistemas kanban, creí en la transparencia de trabajo-en-progreso, la fecha de entrega (throughput) y la calidad, porque entendí que ello genera confianza con los clientes y con la alta dirección. Yo proveía transparencia en cuanto a dónde se encontraba una solicitud dentro del sistema, cuándo podría ser hecho y qué calidad tenía asociada. También proveía transparencia en cuanto a la ejecución del equipo. Hice esto para que los clientes tuvieran confianza de que estábamos trabajando en su solicitud y con una idea de cuándo sería completado. Adicionalmente, quería educar a la alta dirección en cuanto a nuestras técnicas y ejecución para ganar su confianza en mí como director y en mi equipo como un grupo profesional de ingenieros de software bien formado.

Hay un efecto de segundo orden a partir de esta transparencia que no había predicho. Mientras que la transparencia de solicitudes de trabajo y rendimiento estaba muy bien, la transparencia en cuanto al proceso y como funciona tuvo un efecto mágico. Les permite a todos aquellos involucrados ver los efectos de sus acciones e inacciones. Como resultado, la gente se hizo más razonable. Cambiaron su comportamiento para mejorar la ejecución de todo el sistema. Colaboraron en cambios requeridos en las políticas, personal, niveles de asignación de recursos de personal, etc.

META 8 DISEÑAR UN PROCESO PARA PERMITIR EL SURGIMIENTO DE UNA ORGANIZACIÓN DE “ALTA-MADUREZ”

Para la mayoría de los líderes senior de negocio con los que he hablado, esta meta final realmente representa los deseos y expectativas para sus negocios

y sus organizaciones de desarrollo de tecnología. Buscan previsibilidad más que nada, acoplada con agilidad de negocio y buen gobierno.

Los líderes de negocio desean poder hacer promesas a sus colegas en la mesa de comité ejecutivo, a la reunión directiva, a las partes interesadas, a los clientes y al mercado en general y desean poder mantener esas promesas. El éxito a nivel de alto ejecutivo depende en mucho de la confianza y para ello debemos ser confiables. Los altos líderes desean que los riesgos sean gestionados apropiadamente de tal manera que puedan dar resultados previsibles más que otra cosa.

En adición a lo anterior, reconocen que el mundo actual se mueve velozmente y que los cambios suceden rápidamente: nuevas tecnologías llegan; cambios de globalización tanto en los mercados laborales como en los mercados de consumo, generando altas fluctuaciones de demanda (de productos) y oferta (laboral); las condiciones económicas cambian; la competencia cambia sus estrategias y ofertas de mercado y los gustos del mercado cambian conforme la población envejece y se hace más solvente y la clase media se incrementa. Por ello, los líderes de negocio desean que sus organizaciones sean ágiles. Desean responder a los cambios rápidamente y tomar ventaja de las oportunidades.

Detrás de todo esto, desean un buen gobierno. Desean demostrar que los fondos de los inversionistas son utilizados sabiamente. Desean los costos bajo control y desean que los riesgos de su cartera de inversión estén distribuidos óptimamente. Para hacer todo esto les gustaría tener más transparencia en sus organizaciones de desarrollo de tecnología. Les gustaría saber el estado real de sus proyectos y poder ayudar cuando sea apropiado. Desean una organización administrada más objetivamente que reporta hechos con datos, métricas e indicadores; no con anécdotas y evaluaciones subjetivas.

Todos estos deseos equivalen a una organización operando a lo que el SEI define como Nivel de Madurez 4 de su escala de 5 puntos de capacidad y madurez en el CMMI. Los niveles 4 y 5 en esta escala son conocidos como niveles de alta madurez. Muy pocas organizaciones han logrado tales niveles de madurez independientemente de si han buscado o no un método de evaluación estándar formal de CMMI para la mejora de procesos (SCAMPI). No es sorprendente que la mayoría de los altos líderes de las empresas de tecnología grandes estén frustrados por la ejecución de sus equipos de ingeniería de software debido a que la madurez organizacional actual no alcanza el nivel deseado.

Conozca las metas y articule los beneficios

Tenemos ahora un conjunto de metas para nuestro sistema kanban. Necesitamos saber estas metas para poder articularlas porque necesitamos obtener un acuerdo con las partes interesadas dentro de nuestra cadena de valor antes de comenzar con Kanban. Kanban cambiará la manera en la que interactuamos con otros grupos en el negocio si queremos que las partes interesadas acepten los cambios.

Lo que sigue es una guía prescriptiva paso a paso para iniciar un sistema kanban con una cadena de valor simple en su organización. Esta guía fue desarrollada basándose en experiencias reales y validada por varios adoptantes iniciales de Kanban, los cuales siguieron de manera aproximada estos pasos y fueron exitosos y tomando también en cuenta a quienes reconocieron que su falla parcial habría sido prevenida si esta guía hubiera estado disponible en ese entonces.

Esta guía es proporcionada en parte para atraer atención a la diferencia entre Kanban y métodos tempranos de desarrollo ágil. Kanban requiere de un compromiso colaborativo con una cadena de valor más amplia y con la dirección media (y tal vez alta) desde el principio. Una adopción unilateral de Kanban a nivel de raíz tendrá un éxito limitado y entregará beneficios limitados al negocio si primero no construye un consenso de directores externos para el equipo inmediato.

Se me ha indicado que este conjunto de pasos puede parecer atrevido y algunas personas han comentado que si hubiesen leído esto probablemente habrían hecho Kanban a un lado por entero. Tengo la esperanza de que el amplio alcance de este libro haya explicado cómo tomar cada uno de estos pasos y le haya proporcionado consejos útiles obtenidos de la experiencia en el campo.

Pasos para comenzar

1. Acordar un grupo de metas para introducir Kanban.
2. Mapear la cadena de valor (la secuencia de todas las acciones que la organización de desarrollo lleva a cabo para satisfacer las solicitudes de los clientes/partes interesadas). (Ver capítulo 6).
3. Definir algún punto donde desea controlar la entrada. Definir lo que son los niveles anteriores a ese punto y sus partes interesadas (explicado en el capítulo 6). Por ejemplo, ¿desea controlar los requerimientos que llegan al equipo de diseño antes de producción? Las partes interesadas de nivel anterior pueden ser los directores de producto.
4. Definir un punto de salida a partir del cual usted no desea tener control. Definir lo que son los niveles posteriores y sus partes interesadas (explicado en el capítulo 6). Por ejemplo, probablemente usted no necesita controlar la entrega del producto.

5. Definir un conjunto de tipos de ítems de trabajo basados en los tipos de solicitudes de trabajo que llegan de las partes interesadas del nivel superior (explicado en el capítulo 6). ¿Tiene tipos de ítems que son sensibles al tiempo y otros que no lo son? Si es así, entonces puede requerir algunas clases de servicios (explicado en el capítulo 11).
6. Analizar la demanda por cada tipo de ítem de trabajo. Observe la tasa de llegada y su variación. ¿Es la variación estacional o basada en evento? ¿Qué riesgos están asociados con este tipo de demanda? ¿Se debe diseñar el sistema para tratar demanda típica o por picos? ¿Cuál es la tolerancia por entregas tardías o no confiables de este tipo de trabajo? Genere un perfil de riesgo para la demanda. (Explicado en el capítulo 6).
7. Reúnase con las partes interesadas de niveles anteriores y posteriores—puede que sea una reunión grande, o un conjunto de reuniones pequeñas. (Explicado a fondo más adelante en éste capítulo).
 - a. Discuta las políticas alrededor de la capacidad del pedazo de la cadena de valor que desea controlar y llegue a un acuerdo en el límite de WIP (explicado en el capítulo 10).
 - b. Discuta y acuerde un mecanismo de coordinación de entrada con los socios de niveles anteriores, tal como una reunión de priorización regular (explicado en el capítulo 9).
 - c. Discuta y acuerde un mecanismo de coordinación de entrega/lanzamiento con los socios de niveles posteriores, tal como una entrega regular de software (explicado en el capítulo 8).
 - d. Puede necesitar introducir el concepto de diferentes clases de servicio para solicitudes de trabajo (explicado en el capítulo 11).
 - e. Acuerde un objetivo para el tiempo de entrega de cada clase de servicio de los ítems de trabajo. A esto se le conoce como acuerdo de nivel de servicio (SLA; en inglés: *service-level agreement, SLA*) y está explicado en el capítulo 11.
8. Genere una pared de tarjetas para rastrear la cadena de valor que está controlando (explicado en los capítulos 6 y 7).
9. Opcionalmente, cree un sistema electrónico para rastrear y reportar lo anterior (explicado en los capítulos 6 y 7).
10. Concuerde con el equipo el tener una reunión de pie todos los días en frente del tablero o pared de tarjetas e invite a las partes interesadas de los niveles anteriores y posteriores, pero no demande su involucramiento (explicado en el capítulo 7).

11. Concuerde tener reuniones regulares de revisión de operaciones para un análisis retrospectivo del proceso. Invite a las partes interesadas de los niveles anteriores y posteriores, pero no demande su involucramiento (explicado en el capítulo 14).
12. Eduque al equipo en cuanto al nuevo tablero, límites de WIP y el sistema de arrastre. Nada más en su mundo debe haber cambiado. Las descripciones de trabajo son las mismas. Las actividades de trabajo son las mismas. Las entregas son las mismas. Los artefactos son los mismos. Los procesos no han cambiado, a excepción de pedirles que acepten un límite de WIP y que tomen trabajo basado en la política de clase de servicio en lugar de recibirla de manera empujada.

Kanban impacta con un tipo distinto de ganga

Kanban requiere que el equipo de desarrollo de software impacte a los socios empresariales con un tipo distinto de ganga. Para entender esto debemos entender primero las alternativas de uso común.

La gestión tradicional de proyectos hace promesas basadas en la triple restricción de alcance, itinerario y respuesta. Después de alguna estimación y planificación, el presupuesto es asignado para proveer recursos y se llega a un acuerdo en el alcance de requerimientos e itinerario.

Mientras tanto, la gestión ágil de proyectos no tiene un compromiso rígido firmado con sangre. Puede haber una fecha de entrega acordada para algunos meses en el futuro pero el alcance preciso nunca se indica. Alguna definición de alcance de alto nivel puede ser establecida pero los detalles finos nunca son concretados. Pudo haberse acordado un presupuesto (la tasa de consumo financiero mensual) con el fin de proveer un monto fijo de recursos. El equipo de desarrollo ágil procede de manera iterativa, entregando incrementos de funcionalidad en iteraciones pequeñas del *timebox* (o *sprints*), las cuales son habitualmente de una longitud de entre una y cuatro semanas. Al principio de cada una de estas iteraciones se lleva a cabo alguna planificación y estimación y se hace un compromiso. A menudo, el enfoque es priorizado y se entiende que si el equipo no puede cumplir con el compromiso éste se baja y la fecha de entrega permanecerá constante. A nivel de iteración (o *timebox*) el desarrollo ágil parece muy similar a la gestión tradicional de proyectos. La diferencia clave es el entendimiento explícito de que el alcance será bajado si algo se tiene que sacrificar; mientras que un director de proyectos tradicional elegiría retrasar el itinerario, agregar recursos, reducir el alcance o una combinación de ellos.

Kanban nos impacta con un tipo distinto de ganga. Kanban no busca hacer una promesa y comprometerse basándose en algo que es incierto. Una implementación típica de Kanban involucra el acuerdo de que habrá una entrega regular desoftware que funcione de alta calidad—tal vez cada dos semanas. A las partes interesadas externas se les ofrece

transparencia total en el funcionamiento del proceso y si lo desean también podrán tener visibilidad diaria del progreso. Igualmente se les ofrece oportunidades frecuentes para seleccionar los ítems nuevos más importantes para ser desarrollados. La frecuencia de este proceso de selección puede ser más alta que la tasa de entrega—habitualmente una vez por semana, aunque algunos equipos han logrado selección bajo demanda o tasas muy frecuentes, tales como diarias o dos veces por semana.

El equipo ofrece hacer su mejor trabajo y entregar el mayor monto de software que funcione posible; y hacer esfuerzos continuos para incrementar la cantidad, frecuencia y tiempo de entrega. Además de ofrecerle al negocio increíble flexibilidad para seleccionar ítems para procesamiento en cantidades muy pequeñas, el equipo también puede ofrecerle al negocio flexibilidad adicional de priorización e importancia ofreciendo varias clases de servicio para el trabajo. Este concepto está explicado en el capítulo 11.

Kanban ofrece un compromiso sobre cierto monto de trabajo a liberar en cierto día. Ofrece un compromiso sobre los acuerdos de nivel de servicio para cada clase de servicio, respaldado con el compromiso de entregas regulares confiables, transparencia, flexibilidad de priorización y procesamiento y continuo mejoramiento de la calidad, el throughput, la frecuencia de entrega y el tiempo de entrega. Kanban ofrece un compromiso sobre un nivel de servicio, equilibrando el riesgo mediante agregación a través de un monto mayor de ítems. Un sistema kanban diseñado propiamente ofrece un compromiso sobre cosas que los clientes realmente valoran. En intercambio el equipo pide un compromiso a largo plazo de parte de los clientes y los socios de la cadena de valor: un compromiso de tener una relación continua de negocios en la cual el equipo de desarrollo de software se esfuerza constantemente para mejorar el nivel de servicio a través de la mejora de calidad, el throughput, la frecuencia de entrega y el tiempo de entrega. Debido a que el cliente reconoce esa relación continua y a largo plazo y si está dispuesto a medir el nivel de servicio en lugar de mantener al equipo en precisión con los ítems, se puede hacer que el sistema funcione.

El enfoque tradicional para generar el compromiso alrededor del alcance, el itinerario y el presupuesto son indicativos de una operación unitaria. Implica que no existe una relación continua y si un nivel bajo de confianza.

El enfoque de Kanban se basa en la noción de que el equipo permanecerá junto y comprometido en una relación con el proveedor por un largo período de tiempo. El enfoque de Kanban implica mucho negocio recurrente. Implica un compromiso hacia una relación y no hacia un trabajo. Kanban implica que es deseado un alto nivel de confianza entre el equipo de software y sus socios de cadena de valor. Implica que cada uno crea que está formando una asociación a largo plazo y se desea que esa relación sea efectiva.

Un compromiso de Kanban le pide a cada uno en la cadena de valor que tenga cuidado sobre el rendimiento del sistema — que tenga cuidado en la cantidad y la calidad del software entregado, de la frecuencia de entrega y del tiempo de entrega. Kanban les pide a los socios de la cadena de valor que se comprometan al concepto de la agilidad del negocio

real y que acuerden trabajar de manera colaborativa para que eso suceda. Esto diferencia Kanban de manera significativa con respecto a enfoques ágiles iniciales de desarrollo de software.

Al tomar el tiempo necesario para establecer la ganga de Kanban entre las partes interesadas de los niveles anteriores y posteriores del proceso se obtiene un compromiso implícito de rendimiento a nivel del sistema. Se genera el fundamento para una cultura de mejora continua.

Logrando la ganga de Kanban

Una pieza crítica para la implementación exitosa de Kanban es la negociación inicial de este tipo distinto de ganga. Lo que sucede durante estas negociaciones iniciales es el establecimiento de las reglas del juego colaborativo de desarrollo de software que se jugará en adelante. Es vital que los socios de la cadena de valor estén involucrados en el establecimiento de estas reglas porque será necesario que ellos se apeguen a tales reglas si el juego se va a jugar de manera justa y el resultado va a reflejar las metas y la intención.

El paso 7 de nuestro proceso de 12 pasos para introducir Kanban sugiere que nos reunamos con las partes interesadas de los niveles anteriores de la cadena de valor, tales como de marketing o la gente de negocio quienes proveen requerimientos y con las partes interesadas de los niveles posteriores, tales como operación de sistemas y equipos de entrega, u organizaciones de ventas y entrega. Necesitamos acordar con ellos las políticas alrededor del WIP, la priorización, la entrega, las clases de servicio y los tiempos de entrega. El conjunto de políticas acordadas con esos socios definirán las reglas de nuestro juego colaborativo de desarrollo de software. Es difícil tratar cada uno de los cinco elementos de manera aislada porque están esencialmente interrelacionados, por lo que entendemos que debemos establecer políticas alrededor de cada elemento. Las negociaciones serán circulares en su naturaleza conforme los participantes iteran sobre las opciones. Por ejemplo si el objetivo para un tiempo de entrega propuesto es aceptable, puede ser posible introducir una clase de servicio diferente que ofrece un tiempo de entrega más corto para ciertos tipos de solicitud de trabajo. Los cinco elementos – WIP, Priorización, Entrega, Clase de Servicio y Tiempo de Entrega – proveen palancas que pueden arrastrarse para afectar la ejecución del sistema. La habilidad yace en saber cómo jalar esas palancas y cómo negociar opciones para elaborar un acuerdo que funcione efectivamente.

Límites de WIP

Conocí a un director de desarrollo en Dinamarca que me dijo que sus desarrolladores trabajan en siete y media tareas simultáneamente en promedio. Esto es claramente indeseable.

Me pregunto si cualquier persona realmente cree que este nivel de multitareas es apropiado. Si yo estuviera en su lugar utilizaría este hecho como un punto de partida para mis negociaciones. Iniciaría la conversación diciendo que en promedio los miembros del equipo están trabajando en siete y media cosas en paralelo. Indicaría lo que esto le hace al tiempo de entrega y a la previsibilidad, e invitaría a mis colegas y otras partes interesadas a sugerir cuál sería un mejor número. Algunos sugerirían que un ítem por persona es la mejor idea; y puede que sea, pero es una elección muy agresiva. ¿Qué tal si algo se bloquea? ¿No sería bueno tener una alternativa a tomar? Probablemente otra persona sugeriría que dos cosas en paralelo es la respuesta correcta. Algunos pueden argumentar tres; y el rango de sugerencias puede que caiga entre uno y tres. Si el equipo tiene diez desarrolladores y obtiene un consenso de un máximo de dos cosas en proceso por persona, entonces tiene un acuerdo de un límite de WIP de 20 para el equipo de desarrollo.

Hay algunas alternativas. Probablemente desea equipos que trabajen en pares de programadores, por lo que dos cosas por pareja con diez desarrolladores significarán un límite de WIP de diez. Alternativamente podría estar utilizando un método altamente colaborativo tal como Desarrollo Basado en Características o Tripulaciones de Característica (en inglés: *Feature Crew*), en el que pequeños equipos de hasta cinco o seis personas trabajan en un MMF, historias de usuario o lotes de características (tales como FDD), conocido como un Paquete de Trabajo del Programador en Jefe (en inglés: *Chief Programmer Work Package –CPWP*). Un equipo de FDD puede acordar limitar los CPWPs a tres de un equipo de diez desarrolladores. (Normalmente un CPWP se optimiza para hacer más eficiente el desarrollo basado en un análisis de arquitectura del dominio y contiene entre cinco y quince funciones desmenuzadas).

Llevamos entonces una conversación sobre el límite del WIP con las partes interesadas. Hicimos esto discutiendo cuál sería una expectativa razonable para multitareas y relacionándolo con la confiabilidad de las expectativas del tiempo de entrega. Hacer que nuestros socios concuerden con los límites de WIP es un elemento vital. Aunque podríamos declarar los límites de WIP unilateralmente, involucrando a las partes interesadas y formando un consenso estableceremos un compromiso a las reglas de nuestro juego colaborativo. En algún punto en el futuro este compromiso será invaluable. Llegará el día en que nuestros socios nos pidan que tomemos algún trabajo adicional lo cual será porque algo es importante y valioso. Sus razones y motivos serán genuinos y cuando lo hagan seremos capaces de responder pidiéndoles que reconozcan que hemos acordado en un límite de WIP. Es probable que nuestro sistema esté lleno y aceptando otro ítem, cualesquiera sea su importancia, romperá ese límite, por lo que nuestra respuesta podría ser:

“Si, nos encantaría aceptar este nuevo trabajo pues nos damos cuenta que es muy importante para ustedes. Igualmente, ustedes saben y entienden que acordamos en un límite de WIP. Ustedes fueron parte de esa decisión y entienden por qué lo hicimos. Deseamos ser capaces de procesar solicitudes de manera confiable y a tiempo. Para tomar su solicitud

necesitamos poner algo a un lado. ¿Cuál de los ítems actualmente en progreso prefieren que hagamos a un lado para poder iniciar el nuevo ítem?”.

Priorización

También deseamos acordar un mecanismo para aprovisionar la cola. Normalmente estamos buscando un acuerdo para tener una reunión regular de aprovisionamiento y un mecanismo sobre como seleccionar nuevo trabajo. Podemos tener esta conversación preguntando, “si te fuéramos a hacer una pregunta muy simple tal como, ‘¿cuáles dos cosas necesitas que te sean entregadas dentro de 42 días a partir de ahora? ¿qué tan seguido te podrías reunir con nosotros para tener tal discusión?’ Esperaríamos que la reunión no durará más de 30 minutos”. Debido a que está ofreciendo hacer la reunión extremadamente enfocada y está haciendo una pregunta muy directa y sugiriendo que el compromiso de tiempo es mínimo, normalmente encontrará que los socios de los niveles anteriores estarán bastante deseosos de colaborar. No es raro acordar una reunión semanal. Realizarla más a menudo es común en dominios que se mueven muy rápido tales como medios de comunicación donde los ciclos de entrega pueden ser muy frecuentes.

Entrega/entrega

Ahora debemos acordar algo similar con los socios de los niveles posteriores. Una cadencia de entrega que tiene sentido es muy específica de un dominio o situación. Si se trata de software basado en WEB, tenemos que entregarlo en una granja de servidores. La entrega involucra copiar archivos y probablemente actualizar un esquema de base de datos para entonces migrar los datos de una versión del esquema a otra. Esta migración de datos probablemente tendrá su propio código y tomará su propio tiempo de entrega. Para calcular el tiempo de entrega total necesitamos factorizar cuántos archivos copiar en cuántos servidores, cuánto tiempo toma bajar los sistemas adecuadamente y reiniciarlos, cuánto toma migrar los datos y otros. Algunas entregas pueden tomar minutos, otras horas o inclusive días. En otros dominios puede ser necesario manufacturar medios físicos tales como DVD's, embalarlos en cajas y distribuirlos por canales de medios físicos a distribuidores, minoristas o clientes corporativos. Puede haber otros elementos involucrados tales como la impresión de manuales físicos o entrenar al personal de ventas y soporte. Podemos tener que crear un programa de entrenamiento para esas personas.

Por ejemplo, en 2002, formé parte de una entrega para la primera de una serie de actualizaciones de la red de telefonía móvil de Sprint PCS. Esta primera actualización en campo de la tecnología 3G se le llamó 1xRTT. Fue lanzado al mercado por PCS Vision. El lanzamiento involucró la entrega de alrededor de 15 nuevos teléfonos celulares con un

objetivo de 16 nuevas características que utilizaban la capacidad de datos de alta velocidad de la red. Sprint tenía una red de vendedores al menudeo a través de Estados Unidos que empleaba a 17,000 personas. Tenía un número similar de personas en los Call Center's que tomaban llamadas de usuarios para atenderlos. Tanto el canal de ventas al menudeo como los asociados de atención al cliente tenían que ser entrenados para soportar la entrega del nuevo servicio. A manera de broma sugerí que la mejor manera de hacer esto sería apagar todo por dos días, volar a todos a Kansas City por una noche y rentar el estadio de los Jefes de Kansas City, donde podríamos darles una presentación de PowerPoint en las pantallas gigantes del estadio. Esto habría sido la forma más eficiente pero era totalmente inaceptable por varias razones. Nuestros clientes no aceptarían prescindir del servicio por 48 horas mientras entrenábamos a nuestros operadores sobre la siguiente generación de nuestra tecnología. Y perder dos días de ventas de parte del canal de menudeo no habría ayudado a los objetivos de ingreso anual.

Se creó y liberó un programa de entrenamiento que llamamos entrena-al-entrenador. También se creó un programa para entrenar al personal de menudeo regional pues era similar al de los Call Center's. Los entrenadores fueron enviados al campo por seis semanas para capacitar pequeños grupos de personas conforme terminaban sus turnos de trabajo. El costo de liberar el entrenamiento fue enorme. El compromiso de tiempo—6 semanas—fue significativo y la vida media de la capacitación en la memoria de la fuerza de trabajo, también fue de 6 semanas. Si hubiéramos perdido la ventana de entrega para ese nuevo servicio, el entrenamiento se habría tenido que repetir y la entrega habría tenido que ser retrasada por lo menos seis semanas más.

Si nuestro dominio es como una red de telefonía, usted sabe que la cadencia de entrega será infrecuente. Cuando los costos de transacción para hacer una entrega incluyen seis semanas de capacitación, liberar con una mayor frecuencia a la anual, es prohibitivo.

El resultado que usted desea es la cadencia más frecuente de entrega que tenga sentido. Por lo que comience preguntado, “¿si le damos código de alta calidad con defectos mínimos y con notificación adecuada, transparencia en la complejidad y confiabilidad de entrega, qué tan seguido podría razonablemente entregarlo a producción?”. Esto provocará alguna discusión alrededor de las definiciones que usted ha utilizado y se requerirá alguna reiteración de la seguridad. Sin embargo, usted debe insistir para obtener un resultado que maximice la agilidad del negocio sin sobre-estresar ninguna parte del sistema.

Tiempos de entrega y clases de servicio

Refiriéndonos a nuestra conversación sobre tiempo de entrega nos ayuda contar con datos históricos sobre rendimiento pasado. Idealmente, deseamos tener datos sobre el tiempo de entrega y el tiempo de tarea de ingeniería. En el ejemplo de Microsoft del capítulo 4 sabíamos que los tiempos de entrega eran de alrededor de 125 días para defectos de severidad 1

y de 155 días para las otras severidades. Lo primero que nos debe impactar a este respecto es que hay dos clases de servicio. Los defectos de severidad 1 habían recibido alguna forma de tratamiento preferencial históricamente. Puede que nunca haya existido formalidad alguna alrededor de esto pero el resultado neto es que los defectos de severidad uno eran procesados más rápido.

Esto nos permite ofrecer dos clases de servicio distintas. Podemos sugerirle a las partes interesadas externas que adoptaremos dos clases de servicio y tendremos para cada uno de ellos objetivos separados de tiempos de entrega.

También sabemos, a partir de datos históricos, que el esfuerzo de ingeniería promedio era de 11 días y que el tiempo mayor era de 15 días. Elegimos sugerir un tiempo de entrega de 25 días a partir del tiempo de selección de la cola de entrada. No había mayor ciencia al respecto. Imagíñese el efecto psicológico de esto. El negocio estaba acostumbrado a una ejecución de entre cuatro y cinco meses y les acabábamos de ofrecer 25 días. La diferencia está en que les ofrecimos 25 días de tiempo de espera sin incluir el encolado inicial y los 155 días de tiempo de entrega que lo involucraban. Aun así, sonó como una mejora fantástica por lo que no es sorprendente que la gente de negocios haya aceptado.

Existen otras alternativas. Podría tomar los datos históricos del esfuerzo de ingeniería y ponerlos en una gráfica de control estadístico de proceso. Esto le daría un límite de control máximo (o 3-sigma). Puede entonces desear amortiguar el límite superior con un pequeño monto de seguridad que absorberá las variaciones externas. Pero si hace eso, debe ser transparente con los socios y mostrarles como está calculando los números.

Otra alternativa sería preguntar qué nivel de respuesta necesita realmente el negocio. Esto puede lograrse mejor dentro del contexto de un conjunto de clases de servicio. Por ejemplo, si el negocio nos contesta, “necesitamos entrega en tres días”. Usted podría contestar, “¿necesitamos liberar todo en tres días?”. La respuesta será muy probablemente, “no”. Eso le dará la oportunidad de preguntar por una definición de los tipos de solicitudes para este tipo de trabajo y repita este proceso para el resto del trabajo. El resultado debe ser la estratificación de las solicitudes de trabajo en varias bandas para las cuales se puede crear una clase de servicio. Es posible que cada una de esas bandas contenga trabajo que exhibe la misma forma o función por costo de retraso. El detalle alrededor de la creación de clases de servicio y el concepto de funciones de costo de retraso está explicado a detalle en el capítulo 11.

El objetivo del tiempo de entrega que está acordando para cada clase de servicio debe ser presentado como un objetivo en lugar de como un compromiso. Usted se comprometerá a hacer lo mejor para lograr el tiempo objetivo y para reportar la ejecución de la fecha límite contra el tiempo de entrega objetivo en el acuerdo de nivel de servicio (SLA) para cada clase de servicio. En algunas situaciones no habrá suficiente confianza para permitir el acuerdo de que el tiempo de entrega en el SLA es un objetivo más que un compromiso. Si no necesita hacer este acuerdo, debe amortiguar el objetivo con un margen de seguridad.

Esto indicará directamente que un bajo nivel de confianza resulta en un costo económico directo.

El criterio de salida de la discusión con su socio es la siguiente: usted tiene un consenso sobre los límites de WIP a lo largo de la cadena de valor; usted tiene un acuerdo sobre la coordinación de priorización y el método a utilizarse; usted tiene un acuerdo similar sobre la coordinación y el método de entrega; usted tiene una definición de un conjunto de acuerdos a nivel de servicio que incluyen un tiempo de entrega objetivo para cada clase de servicio.

Para llevar

- ❖ Existen por lo menos ocho metas posibles para introducir Kanban en su organización.
- ❖ Mejore el rendimiento a través de mejoras de proceso introducidas con resistencia mínima.
- ❖ Libere con alta calidad.
- ❖ Libere en un tiempo de entrega previsible controlando la cantidad de trabajo-en-progreso.
- ❖ De a los miembros del equipo una mejor vida mediante una mejora de equilibrio entre el trabajo y la vida.
- ❖ Provea holgura en el sistema equilibrando la demanda contra el throughput.
- ❖ Proporcione un mecanismo simple de priorización que retrase el compromiso y mantenga las opciones abiertas.
- ❖ Provea un esquema transparente para ver las oportunidades de mejora, permitiendo un cambio hacia una cultura más colaborativa que promueva mejora continua.
- ❖ Luche por un proceso que permita resultados previsibles, agilidad de negocio, buen gobierno y el desarrollo de lo que el Software Engineering Institute llama una organización de alta madurez.
- ❖ Es importante definir sus metas y ser capaz de articular los beneficios de introducir Kanban con el fin de obtener un consenso con las partes interesadas.
- ❖ Siga la guía de doce pasos para iniciar un proceso de Kanban.
- ❖ Kanban impacta una gama diferente con las partes interesadas externas y los propietarios de negocio. Es una gama basada en el supuesto de una relación a largo plazo y un compromiso de la ejecución a nivel de sistema.
- ❖ Incluir a las partes interesadas externas para formar un acuerdo sobre los elementos básicos del sistema kanban los hace colaboradores.
- ❖ Las políticas básicas sobre los límites de WIP, los objetivos de tiempo de entrega, las clases de servicio, la priorización y la entrega representan las reglas del juego colaborativo del desarrollo de software.
- ❖ Involucrar a las partes interesadas externas como colaboradores para acordar sobre las reglas del juego, permitirá un comportamiento colaborativo más adelante cuando el sistema es puesto bajo estrés.

❖ PARTE CUATRO ❖

HACIENDO MEJORAS

❖ CAPÍTULO 16 ❖

Tres tipos de oportunidad de mejora

Los capítulos 6 al 15 describen como construir y operar un sistema Kanban y como adoptar el enfoque de Kanban para la gestión y mejoramiento de cambios. El resto del libro describe como reconocer oportunidades de mejora, qué hacer al respecto y cómo escoger entre ellos.

El capítulo 2 identifica las cinco propiedades centrales que usted esperaría encontrar en una organización que utiliza Kanban. La quinta propiedad describe como los modelos son usados para identificar, evaluar y dirigir oportunidades de mejora. Hay muchos modelos posibles. Éste capítulo se enfoca en tres modelos comunes y algunas de sus variaciones: la Teoría de Restricciones y sus Cinco Pasos de Focalización; un subconjunto de ideas del Pensamiento Lean que identifica actividades con desperdicio como costos económicos; y algunas variantes que se enfocan en entender y reducir variabilidad. Otros modelos son posibles. La comunidad ya está experimentando con modelos tales como la Teoría de Opción Real y Gestión de Riesgos. Lo que damos a continuación son ejemplos. Son un punto de partida. Los invito a que los adopten, pues sé que funcionan y los animo a que expandan su manera de pensar y a que busquen una variedad más amplia de modelos que les permitan facultar equipos para generar mejoras.

Cuellos de botella, eliminación de desperdicio y reducción de variabilidad

Cada uno de estos modelos para mejora han sido explorados y desarrollados enteramente dentro de su propio cuerpo de conocimiento. Cada uno tiene su propia escuela de pensamiento sobre mejora continua. Con Kanban, he elegido sintetizar los tres y proveer una visión general sobre cómo reconocer estas oportunidades de mejora y detalles sobre cómo implementar mejoras utilizando cada modelo. Cada una de estas tres escuelas de pensamiento sobre mejora continua descrita abajo tiene su propio grupo de líderes de pensamiento. Su empresa puede suscribirse a una o más de estas escuelas. Puede ser una gran ventaja el poder mostrar como las técnicas de Kanban pueden proveer oportunidades de mejora dentro de lo que su organización prefiere. Saber que tiene un conjunto amplio de paradigmas de mejora y herramientas a escoger le debe proveer una flexibilidad mayor para hacer el cambio.

Aquellos ampliamente familiarizados con metodologías de mejora continua pueden escoger saltarse el resto de éste capítulo y pasar directamente al capítulo 17. Quienes deseen una visión general de los métodos disponibles y un antecedente literario e histórico, pueden encontrar valioso el resto de éste capítulo valioso.

Teoría de Restricciones

La Teoría de Restricciones fue desarrollada por Eli Goldratt y fue publicada por primera vez en su novela de negocios, *La Meta* (en inglés: *The Goal*), en 1984. Durante los últimos 25 años, *La Meta* ha pasado por varias revisiones y el marco teórico conocido como los “Cinco Pasos de Focalización” se ha vuelto más obvio en las ediciones más recientes.

Los Cinco Pasos de Focalización son la base para la mejora continua en la teoría de limitaciones. Se le conoce como un POOGI (proceso de mejora continua. En inglés: *Process of OnGoing Improvement*). La Teoría de Restricciones (TDR) está llena de acrónimos. Extrañamente, los Cinco Pasos de Focalización son la excepción. No está abreviado como “FFS” (en inglés: *Five Focusing Steps*).

En los 90, la Teoría de Restricciones evolucionó un método para un análisis de causa raíz y gestión de cambios conocido como el Proceso de Pensamiento (en inglés: *Thinking Process—TP*). La razón de esta mejora fue el descubrimiento dentro de la comunidad de TDR de que su restricción para lograr mejoras en los clientes eran la gestión de cambios y la resistencia al cambio.

Parecía ser que los Cinco Pasos de Focalización solamente funcionaban bien para problemas de flujo y que muchos retos del área de trabajo no cabían bien dentro del paradigma de flujo. Por lo que el TP evolucionó. El programa de entrenamiento y calificación técnica para los consultores de TDR cambió de ser una clase sobre el uso de los Cinco Pasos de

Focalización y sus aplicaciones, tales como Drum-Buffer-Rope, a una clase sobre TP. Por lo tanto cuando muchas personas en la comunidad de TDR se refieren a TDR de hecho se están refiriendo al TP y no a los Cinco Pasos de Focalización. Al atender conferencias sobre TDR observé que el uso de los Cinco Pasos de Focalización dentro de la comunidad de TDR se ha convertido en un arte muerto.

Por lo que he observado, la comunidad de TDR ha tendido a aceptar paradigmas conforme se han establecido, en lugar de retarlos. Por ende, la solución de TDR para gestión de proyectos, conocido como Cadena Crítica (en inglés: *Critical Chain*) evolucionó alrededor del paradigma de gestión de proyectos establecido como la tripe restricción (alcance, presupuesto e itinerario) y el modelo de la gráfica de dependencia para programar las tareas en el proyecto. Nadie retó ese modelo establecido hasta el momento en que publiqué mi primer libro, *Agile Management for Software Engineering*, en el que sugerí que era mejor modelar proyectos como un problema de cadena de valor y flujo y aplicar los Cinco Pasos de Focalización. Haciendo esto se hacia posible utilizar todo el cuerpo de conocimiento Lean, basado en flujo y sintetizarlo con el enfoque en cuellos de botella de los Cinco Pasos de Focalización. La síntesis del TDR con Lean permitió mejoras en el rendimiento organizacional y rendimiento de proyectos y puso el fundamento para el surgimiento de Kanban.

He argumentado que cualquier proceso de flujo de trabajo que involucra la división de valor puede ser definido como una cadena de valor; y se puede observar que cualquier cadena de valor de hecho tiene flujo. Lean y el sistema de producción de Toyota (En inglés: TPS) están construidos basándose esencialmente en esa asunción. Si cualquier cadena de valor tiene flujo, entonces los Cinco Pasos de Focalización pueden ser aplicados a él. Por ende los Cinco Pasos de Focalización son un POOGI perfectamente satisfactorio y el TP no se requiere a menos que se lo esté utilizando como una herramienta de gestión de cambios. Personalmente no he desarrollado ninguna afinidad con TP. Mi herramienta de gestión de cambios preferida es Kanban, como este texto lo demuestra.

Los cinco pasos de focalización

Los Cinco Pasos de Focalización son una fórmula simple para un proceso de mejora continua. Establece que:

1. Identificar la restricción
2. Decidir cómo explotar la restricción
3. Subordinar todo lo demás en el sistema a la decisión hecha en el paso 2
4. Elevar la restricción
5. Evitar la inercia; identificar la siguiente restricción y regresar al paso 2

El paso 1 nos pide que encontremos un cuello de botella en nuestra cadena de valor.

El paso 2 nos pide que identifiquemos el throughput potencial de ese cuello de botella y lo comparemos con lo que está sucediendo actualmente. Como verá, el cuello de botella raramente o nunca está trabajando a capacidad completa. Por lo que debe preguntarse, “¿qué se necesita para obtener el potencial completo de este cuello de botella? ¿qué necesitamos cambiar para que eso suceda?”. Esta es la parte de “decidir” del paso 2.

El paso 3 nos pide que hagamos los cambios necesarios para implementar las ideas del paso 2. Esto puede involucrar hacer cambios adicionales en otras partes de la cadena de valor con el fin de obtener la capacidad máxima del cuello de botella. Esta acción de maximizar la capacidad del cuello de botella es conocida como “explotación del cuello de botella”.

El paso 4 sugiere que si el cuello de botella está operado a su máxima capacidad y aún no produce el suficiente throughput, su capacidad debe ser mejorada con el fin de incrementar el throughput. El paso 4 nos pide que implementemos una mejora de la capacidad y el throughput lo suficientemente como para que el cuello de botella actual deje de serlo y la restricción en el sistema se mueva a otro lado dentro de la cadena de valor.

El paso 5 requiere que le demos tiempo a los cambios para que se estabilicen y entonces identifiquemos el nuevo cuello de botella en la cadena de valor para entonces repetir el proceso. El resultado es un sistema de mejora continua en el cual el throughput siempre se está incrementando.

Si los Cinco Pasos de Focalización son institucionalizados apropiadamente se habrá logrado una cultura de mejora continua a lo largo de toda la organización.

El capítulo 17 explica cómo identificar y gestionar los cuellos de botella utilizando los Cinco Pasos de Focalización.

Lean, TPS y reducción de desperdicio

Lean emergió en los 90's después del texto seminal, *La máquina que cambió al mundo*, (en inglés: “*The Machine that Changed the World*”), escrita por Womack, Jones y Daniels, el cual describe como trabaja el sistema de producción Toyota (TPS) desde un punto de vista empírico externo. La primera literatura sobre Lean tenía algunas fallas. Falló en identificar la gestión de variabilidad que es inherente del TPS y que fue aprendido y adaptado del Sistema de Conocimiento Profundo de Deming. Lean también cayó víctima de la mala interpretación y la sobre-simplificación. Muchos consultores Lean saltaron al concepto de Eliminación/Reducción de Desperdicio y enseñaron Lean enteramente como un ejercicio de eliminación de desperdicio. En este anti-patrón de Lean todas las actividades de trabajo son clasificadas como que agregan-valor o no-agregan-valor. Las actividades de desperdicio que no agregan valor se subclasifican en desperdicio necesario y desperdicio innecesario.

Las actividades innecesarias son eliminadas y las necesarias son reducidas. Aunque éste es un uso válido de las herramientas Lean para hacer mejoras, tiende a suboptimizar el resultado para la reducción de costo y deja valor en la mesa al no aceptar las ideas Lean de Valor, Cadena de Valor y Flujo.

Kanban le da cabida a todos los aspectos del pensamiento Lean y provee las herramientas para optimizar un resultado de valor a través de un enfoque en la administración de flujo así como en la reducción del costo.

El capítulo 18 explica cómo identificar las actividades de desperdicio y qué hacer al respecto.

Deming y seis sigma

W. Edwards Deming es considerado uno de los tres padres del movimiento de Aseguramiento de Calidad en el siglo XX. Sin embargo, su contribución fue considerada mucho mayor. Evolucionó el uso del Control Estadístico de Proceso (CEP) y lo desarrolló en una técnica de gestión la cual llamó Sistema de Conocimiento Profundo. Su sistema tenía la intención de prevenir a los directores de que tomaran decisiones de baja calidad y que las reemplazaran con mejores decisiones estadísticamente sólidas, objetivas y a menudo contra-intuitivas. Ocasionalmente se le considera a Deming como, probablemente, el científico de gestión más importante del siglo XX y en mi opinión lo tiene bien merecido. Sus contribuciones se extendieron del CEP al Aseguramiento de Calidad y a la Ciencia de la Gestión.

Deming tuvo una influencia significativa sobre la filosofía de gestión japonesa alrededor de mediados del siglo XX y su trabajo en relación al CEP y al Sistema de Conocimiento Profundo son los pilares del TPS.

Mientras que algunos equipos de Kanban altamente maduros tales como el banco de inversión BNP Paribas en Londres ha adoptado CEP, tal sistema de control estánmás allá del alcance de este libro y será tratado en un texto futuro sobre técnicas avanzadas de Kanban.

Sin embargo, los principios para entender la variación de sistemas y las tareas de trabajo que son base del CEP son muy útiles. El predecesor de Deming, Walter Shewhart clasificó la variabilidad en la ejecución de tareas en dos categorías: causa aleatoria y causa asignada. Más tarde Deming las renombró como causa común y causa especial; y en la segunda edición de su libro *La Nueva Economía* admite que esto fue “en gran parte por razones pedagógicas”. No había innovación específica en el cambio de términos. El entendimiento de la variación y cómo impacta el rendimiento, así como desarrollar la capacidad de clasificarla dentro de las dos categorías son habilidades de gestión necesarias. El aprendizaje de las acciones de gestión a tomar adecuadas a decidir basado en el tipo de variación es un aspecto central de un programa de mejora continua.

Tanto Lean como La Teoría de Restricciones dependen en gran medida del entendimiento de variación a fin de permitir mejoras, aún si esas mejoras son tomadas como gestión de cuellos de botella o reducción de desperdicio.

El capítulo 19 explica como reconocer las variaciones de causa especial y las variaciones de causa común y sugiere ideas para su administración apropiada. El capítulo 20 trata aún más al respecto, describiendo como construir una capacidad de gestión de asuntos que responde a variaciones de causa assignable con el objetivo de eliminar tales asuntos lo más rápidamente posible con el fin de mantener el flujo y maximizar la entrega de valor. Nota: sin conocimiento y enfoque sobre la gestión de variabilidad, un enfoque sobre el flujo sería ineffectivo. Lean, sin las ideas de Deming, es Lean sin el entendimiento de variación y, por implicación, es Lean sin enfoque en mantenimiento de flujo. Debido a que la literatura Lean inicial no incluía un entendimiento sobre variación y tampoco referencias al Sistema de Conocimiento Profundo de Deming, es fácil entender la causa raíz del anti-patrón de enseñar Lean únicamente como un proceso de reducción de desperdicio.

Mientras que las ideas de Deming están presentes en el TPS en Japón a nivel del piso de la planta de trabajo, donde el CEP y el Sistema de Conocimiento Profundo fueron empleados para identificar oportunidades de mejora local, otro cuerpo de conocimiento se desarrolló en los Estados Unidos, basado alrededor de las ideas de Deming. Seis Sigma comenzó en Motorola, pero realmente maduró cuando fue adoptado en GE bajo el liderazgo de Jack Welch.

Seis Sigma utiliza CEP para identificar variaciones de causa especial y de causa común y utiliza un proceso similar al descrito por Deming para eliminar variaciones de causa especial a nivel de causa raíz y prevenir que sean recurrentes; adicionalmente, para reducir variación de causa común y hacer del proceso, el flujo de trabajo, o el sistema, más predecibles.

A diferencia del TPS, el cual trata enteramente acerca de iniciativas en el piso de trabajo que son ejecutadas por trabajadores facultados que implementan pequeños eventos kaizen entre los cientos de miles, Seis Sigma lo ha desarrollado como un método de comando y control de baja confianza que tiende a involucrar muchas menos oportunidades de mejora y que son generalmente implementadas a un nivel más estratégico y ejecutadas como proyectos específicos. El líder de proyecto viste su pequeño Cinturón negro (Black Belt) y ha pasado por años de entrenamiento en la metodología para ganarse ese estatus. Debido a que Kanban acepta las ideas de Deming y provee la instrumentación y la transparencia para ver variabilidad y sus efectos, puede ser utilizado para permitir ya sea un programa de mejora tipo kaizen o un programa de mejora tipo Seis Sigma.

Ajustando Kanban a la cultura de su empresa

Si su compañía es una empresa Seis Sigma, Kanban puede ayudarle a ejecutar sus iniciativas Seis Sigma en el software, sistemas, desarrollo de productos u organización de TI. Si su compañía es una empresa Lean, Kanban se ajusta de manera natural. Puede permitir una iniciativa completa de Lean en su software, sistemas, desarrollo de producto u organización de TI. Si su empresa se suscribe y usa la Teoría de Restricciones, Kanban permite tener un programa completo de gestión de restricciones (eliminación de cuellos de botella) en su software, sistemas, desarrollo de productos u organización de TI. Sin embargo, podría necesitar moldear la implementación del sistema de arrastre tal como Drum-Buffer-Rope en lugar de referirse a él como un sistema kanban. Debido a que Kanban se desarrolló a partir de una implementación temprana del Drum-Buffer-Rope, sé que esto funcionará. Sin embargo, una discusión sobre los detalles de cómo modelar el flujo de valor y establecer los límites de WIP para el Buffer y el Rope está más allá del alcance de este texto.

Para llevar

- ❖ Kanban requiere que los modelos sean utilizados para identificar oportunidades de mejora.
- ❖ Kanban apoya por lo menos tres tipos de métodos de mejora continua: Gestión de Restricciones (eliminación de cuellos de botella), Reducción de Desperdicio y Gestión de Variabilidad (así como el CEP y el Sistema de Conocimiento Profundo).
- ❖ Kanban permite la identificación de cuellos de botella y una implementación completa de los Cinco Pasos de Focalización de la Teoría de Restricciones.
- ❖ Kanban permite la visualización de actividades de desperdicio y puede utilizarse para permitir una iniciativa Lean completa dentro del software, sistema, desarrollo de producto u organización de TI.
- ❖ Kanban provee la instrumentación para utilizar la Teoría de Conocimiento Profundo de W. Edwards Deming y el Control Estadístico de Proceso. Puede utilizarse para dirigir una iniciativa de kaizen o una iniciativa Seis Sigma.

❖ CAPÍTULO 17 ❖

Cuellos de botella y disponibilidad instantánea

Washington SR-520 es la carretera que une Seattle con los distritos del noreste llamados Kirkland y Redmond. Es la arteria principal de conexión para los habitantes de las zonas urbanas quienes trabajan en el centro de la ciudad y para empleados de Microsoft; y para otras firmas de alta tecnología ubicadas en esos distritos, tales como AT&T, Honeywell y Nintendo, quienes viven en la ciudad y se trasladan en dirección opuesta todos los días. Durante un total de 8 horas todos los días, el camino es un cuello de botella de tráfico en ambas direcciones. Si usted estuviera en el puente que cruza la carretera en la calle 76 NE en la pequeña zona urbana de Medina (calle arriba de la residencia de Bill Gates en la orilla del lago Washington), durante la tarde y mirara hacia el Este, vería el tráfico en dirección Oeste dirigido hacia la ciudad arrastrándose lentamente colina arriba del Bellevue antes de reducirse a dos carriles para cruzar el puente flotante para entrar a Seattle. La velocidad del tráfico subiendo la colina es de alrededor de diez millas por hora y el flujo es desigual debido a los vehículos que constantemente reducen la velocidad y se detienen. Si cruza la calle y ve hacia el Oeste, hacia los rascacielos del centro de Seattle, la Aguja Espacial y las Montañas Olímpicas a lo lejos, verá el tráfico alejándose uniformemente a casi 50 millas por hora. ¿Qué tipo de magia está sucediendo bajo sus pies tal que la velocidad cambia tan dramáticamente y el flujo se transforma de desigual a uniforme?

Justo antes del puente de pontones el camino se hace más angosto, disminuyendo de tres a dos carriles. El carril derecho de la carretera es un carril de vehículos de ocupación alta (high-occupancy vehicle —HOV) el cual requiere que los vehículos tengan dos o más pasajeros y es frecuentado por los autobuses de servicio público que transportan personas de y hacia la ciudad así como automóviles privados. Cuando estos vehículos se unen al tráfico es suficiente para causar la alteración, reducción y acumulación de tráfico. En las varias millas que preceden al puente hay otros caminos que se unen a la carretera agregando más volumen al tráfico del ya muy congestionado camino en las horas pico. El efecto neto es un flujo muy irregular y velocidades muy bajas.

Desde el punto de vista de la seguridad del tráfico, los planificadores se preocupan por la distancia entre automóviles. Idealmente desean una distancia suficiente para reaccionar a los cambios y detenerse de forma segura si es necesario. La distancia se relaciona con la velocidad y el tiempo de reacción. La “distancia” recomendada entre vehículos es de dos segundos. En lenguaje Lean este es el tiempo takt ideal entre vehículos. Por lo que si tenemos dos carriles y dos segundos entre vehículos el throughput máximo del camino es de 30 vehículos por carril por minuto o 60 vehículos por minuto. Esto es cierto independientemente de la velocidad de los vehículos. Estas reglas se rompen en límites extremos de velocidades muy bajas y velocidades super-excesivas—aquellas mucho más arriba del límite de 50 millas forzadas en la SR-520. Para propósitos prácticos el throughput (al cual se refiere de manera confusa como capacidad en la administración de tráfico) es de 3,600 vehículos por hora.

Sin embargo, si está en el puente y cuenta el número de vehículos pasando bajo él en una tarde típica alrededor de las 5:00 PM, notará que menos de 10 vehículos por minuto cruzan el puente flotante hacia Seattle. ¡A pesar de la alta demanda, el camino está operando a menos de un-quinto de su throughput potencial! ¿Por qué?

El puente de pontones sobre el Lago Washington es un cuello de botella. Todos entendemos el concepto intuitivamente. El ancho del cuello de botella controla el flujo de líquido hacia y desde la botella. Podemos verter rápidamente de un cuello ancho, pero a menudo con mayor riesgo de derrame. Con un cuello angosto el flujo es más lento pero puede ser más preciso. Los cuellos de botella restringen nuestro potencial de throughput; en éste ejemplo, de 60 vehículos por minuto, o 3,600 por hora, a menos de 10 vehículos por minuto, o 600 vehículos por hora.

En general, el cuello de botella en el flujo de un proceso está en cualquier lugar en el que un *backlog* se acumula esperando a ser procesado. En el ejemplo de la SR-520, el *backlog* es una cola de vehículos ocasionalmente hasta Overlake, siete millas al Este. En el desarrollo de software, puede ser cualquier *backlog* o trabajo no iniciado, o trabajo-en-progreso: los requerimientos esperando el análisis, el trabajo analizado esperando por el diseño, el desarrollo y las pruebas; el trabajo probado esperando por el desarrollo, etc.

Como se discutió, la SR-520 entrega solamente alrededor d20% de su potencial en los tiempos pico, cuando más se necesita. Para una explicación completa de esto necesitamos entender como explotar enteramente el potencial de un cuello de botella y el efecto que la variabilidad tiene sobre ese potencial. Estos conceptos están explicados aquí en el capítulo 17 y más adelante en el capítulo 19.

Recursos de capacidad restringida

La SR-520 es un cuello de botella de Capacidad Restringida en el puente de la calle 76 NE. Su capacidad es 60 vehículos por minuto en dos carriles. Lo que nos conlleva a lo siguiente: el camino es de tres carriles, por lo que el tráfico es forzado a juntarse con el fin de cruzar el lago por el puente viejo de pontones que fue diseñado hace 50 años con tan solo dos carriles. En ese entonces era bastante capacidad y el puente no era un cuello de botella. Las zonas urbanas al este eran pequeñas villas y trasladarse del otro lado del lago era raro—y en ese entonces solo hacia la ciudad y no al contrario como ahora es común.

Acciones de elevación

En éste respecto la SR-520, como cuello de botella de capacidad restringida, puede ser similar a una diseñadora de experiencia de usuario en un equipo de software quien es responsable de diseñar todas las pantallas y diálogos con los que el usuario interactúa. Trabaja continuamente pero aún así, su throughput es insuficiente para satisfacer la demanda puesta en ella por el proyecto. La reacción natural de muchos directores en esta situación es contratar a otra persona para ayudar. En la Teoría de Restricciones de Eli Goldratt a esto se le conoce como “elevación de la restricción”—agregar capacidad para remover el cuello de botella.

En nuestro ejemplo de la SR-520, esto sería equivalente a sustituir el puente flotante sobre el lago Washington por un nuevo puente con tres líneas de tráfico en cada dirección. Para mantener todas las cosas igual, debería ser un puente con una línea de vehículos de alta ocupación y una línea para bicicletas, así como dos líneas abiertas a todo tipo de tráfico. De hecho, este es el modo en que el departamento de transporte del estado de Washington está actuando. El puente costará muchos cientos de millones de dólares y llevará una década construirlo. En el momento de escribir esto, la construcción aún no ha empezado.

Sucede que elevando el recurso de capacidad restringida debería ser el último recurso. Incrementar la capacidad de un cuello de botella cuesta tanto tiempo como dinero. Si por ejemplo, tenemos que contratar a otro diseñador debemos conseguir presupuesto para pagarle a esta nueva persona así como el presupuesto para el proceso de contratación, el cual puede incluir cuotas que les paguemos a los agentes por obtener referencias. Alentaremos

el progreso en nuestro proyecto actual mientras revisamos currículums y entrevistamos candidatos. Nuestro recurso más precioso, nuestro diseñador de experiencia de usuario de capacidad restringida tendrá que tomar tiempo fuera del proyecto real para leer currículums, seleccionar candidatos y entrevistarlos. Como resultado su capacidad de terminar diseños se reduce y por ende el throughput de todo el proyecto. Esto es parcialmente la razón por la cual la “ley” de Fred Brooks indica que agregar gente a un proyecto retrasado solamente lo retrasa más. Aunque la observación de Brooke es anecdótica podemos dar actualmente una explicación más científica a este fenómeno, el cual ha sido entendido por la industria de software durante al menos 35 años.

Acciones de explotación/protección

En lugar de saltar inmediatamente a elevación y gastar tiempo y dinero mientras que aleñamos las cosas, es mejor pensar primero en maneras de explotar completamente la capacidad del recurso en el cuello de botella. Por ejemplo, la SR-520 tiene un throughput de tan sólo 20% de su potencial en las horas pico. ¿Qué acciones se pueden tomar para mejorar el throughput? Soñemos por un momento. Si el throughput del camino durante las horas pico llegara a alcanzar su potencial de 3600 vehículos por hora, ¿sería necesario reemplazar el puente existente por uno nuevo? ¿Serían los tiempos de viaje lo suficientemente cortos como para que los que paguen impuestos del estado de Washington (como el autor) preferirían que sus impuestos se usaran en alguna otra cosa más importante? ¿Tales como más libros para las escuelas locales? ¡Probablemente!

¿Cómo podríamos entonces explotar el potencial real del camino? La fuente del problema de hecho está en los humanos conduciendo los vehículos. Sus tiempos de reacción y las acciones que toman son altamente variables. Conforme los vehículos se entroncan del carril de HOV, los vehículos en el carril central necesitan reducir la velocidad y hacer espacio para el auto que está ingresando. Algunos conductores reaccionan más despacio que otros; algunos frenan más vigorosamente que otros y el efecto neto es que el tráfico se hace más lento impredeciblemente. Algunos conductores, molestos por la fluctuación en el carril y su velocidad menor comparada con el carril de la izquierda deciden cambiar de carril. El mismo efecto se repite. Todos los vehículos van más lento, pero la velocidad no afecta el throughput. Lo que es importante es el espacio entre los vehículos. Lo que se desea es un flujo suave de tráfico con un espacio de dos segundos entre vehículos. Sin embargo, el elemento humano significa que los vehículos no se desaceleran o aceleran uniformemente y los espacios son como un acordeón. El tiempo de reacción de los individuos para presionar el acelerador y el freno más el tiempo de reacción de los motores y las transmisiones de los vehículos implican que los espacios se amplían conforme el tráfico se incrementa. La variabilidad en el sistema tiene un alto impacto en el throughput.

Arreglar el problema de la SR-520 nos lleva a un mundo de fantasía en términos de control de vehículos, aunque algunas manufactureras alemanas han experimentado con tales sistemas. Los sistemas utilizan un radar o láseres para juzgar la distancia entre vehículos y mantener el tráfico en movimiento como una caravana uniforme pueden remover la variabilidad que ocurre en la SR-520. Tales sistemas tienen la habilidad de hacer más lentas cadenas enteras de vehículos uniformemente mientras mantienen el espacio entre ellos. Como resultado el throughput de tráfico permanece alto. Sin embargo, eliminar la variabilidad de vehículos privados conducidos por sus ocupantes tiene sus límites. Si desea transporte de baja variabilidad necesita encadenar los vehículos de pasajeros y ponerlos en rieles. Esa es la razón fundamental por la cual la el transporte rápido en masa sobre rieles es más efectivo que la de los automóviles moviendo altas cantidades de personas rápidamente.

La buena noticia es que en nuestra oficina los recursos de capacidad restringida son afectados por variabilidad por la que podemos hacer algo. Hemos hablado mucho en este libro respecto a la coordinación de actividades y los costos de transacción de hacer trabajo de valor agregado. Si tenemos un diseñador de experiencia de usuario de capacidad restringida, podemos buscar mantener a esa persona ocupada trabajando en cosas de valor agregado y minimizar las actividades de valor no agregado (desperdicio) normalmente pedidas a esa persona.

Por ejemplo, tuve un equipo de prueba de capacidad restringida en 2003. Para maximizar la explotación de su capacidad busqué otros recursos holgados y los encontré en el analista de negocio y un director de proyecto. Liberamos al equipo de prueba de las actividades burocráticas tales como completar las hojas de tiempo. También los liberamos de la planificación de proyectos futuros. Le permitimos a los analistas desarrollar planes de prueba para iteraciones futuras y para proyectos mientras que los ingenieros de pruebas se mantenían ocupados efectuando pruebas sobre el trabajo-en-progreso actual.

Otro enfoque mejor, uno que no consideré en ese entonces habría sido generar un perfil de riesgos para los requerimientos que deben ser probados únicamente por el equipo profesional de pruebas. Los requerimientos que no cabían dentro de del criterio podrían ser probados por personas de otras áreas funcionales desempeñando un papel de ingeniero de prueba amateur; por ejemplo, el analista de negocio. Esta técnica de “bifurcación”, utilizando un perfil de riesgos es una muy buena manera de optimizar la utilización de un cuello de botella mientras que continuamos gestionando riesgos en el proyecto.

Una solución a largo plazo podría haber sido invertir fuertemente en la automatización de pruebas. La palabra clave es “invertir”. Si utiliza ese término entonces está hablando sobre una acción de elevación. Agregar recursos no es la única manera de elevar capacidad. La automatización es una buena estrategia natural de elevación. La comunidad de desarrollo de software ágil ha hecho mucho para motivar la automatización de pruebas durante la última década. Como regla general considere la automatización como una estrategia general de elevación. Sin embargo, un efecto secundario de la automatización es que también

reduce variabilidad: tareas y actividades repetibles son repetidas con precisión digital. Por lo que la automatización reduce variabilidad en el proceso y ayuda a mejorar la explotación de la capacidad de otro cuello de botella.

La siguiente forma de asegurar máxima explotación de nuestro diseñador de experiencia de usuario de capacidad restringida es asegurar que ella siempre esté teniendo progreso en el trabajo actual. Si la diseñadora de experiencia de usuario reporta que está bloqueada por alguna razón externa, el director de proyecto y, de ser necesario, el equipo entero debe enjambrarse en el asunto y resolverlo. Una capacidad organizacional fuerte en la identificación, escalamiento y resolución de asuntos es esencial para la explotación eficiente de cuellos de botella de capacidad restringida.

Si hay varios asuntos bloqueando el trabajo actual, entonces los asuntos que están impiadiendo al recurso de capacidad restringida, en este caso nuestra diseñadora de experiencia de usuario, debe tener la prioridad más alta. Para la administración efectiva y de alto rendimiento de asuntos es por lo tanto necesario saber la ubicación del recurso de capacidad restringida y darle prioridad cuando sea requerido.

La transparencia en el sistema kanban ayudará a incrementar la conciencia tanto de la ubicación de los recursos de capacidad restringida (cuello de botella) así como el impacto de cualquier asunto que impide el flujo en ese punto en el sistema. Con todos los involucrados en el proyecto conscientes del impacto a nivel de sistema de un impedimento en el cuello de botella, el equipo se enjambrará con gusto en el problema para resolverlo. La alta dirección y las partes interesadas externas con un alto interés de que la entrega se realice a tiempo también darán parte de su tiempo más libremente cuando entiendan el valor de ese tiempo y el impacto que proveerá una solución rápida del asunto.

Por ende, desarrollar una capacidad organizacional de transparencia rastreando y reportando los proyectos utilizando un sistema kanban es crítico para mejorar el rendimiento. La transparencia nos lleva a visibilidad tanto de cuellos de botella como de impedimentos y consecuentemente a la explotación mejorada de la capacidad disponible para hacer trabajo valioso mediante un equipo enfocado en mantener el flujo.

Una técnica más que es comúnmente utilizada para asegurar la explotación máxima de un recurso de capacidad restringida es asegurar que el recurso nunca estará holgado. Sería un desperdicio terrible si el recurso de capacidad restringida se dejara sin trabajo por hacer debido a un problema inesperado en niveles anteriores del proyecto; por ejemplo, un analista de requerimientos toma varias semanas fuera del trabajo debido a una emergencia médica familiar. Repentinamente la restricción se movió. O probablemente una sección grande de requerimientos es retomada por el negocio el cual ha hecho un cambio estratégico. Mientras el equipo espera que se desarrolle los nuevos requerimientos, el diseñador de experiencia de usuario está holgado. ¿Qué tal si las actividades de niveles anteriores son de naturaleza altamente variable? Esto es algo común con la solicitud y desarrollo de requerimientos. Por ende, la tasa de llegada de trabajo a efectuar puede ser irregular.

Puede haber muchas razones por las cuales el recurso de capacidad restringida puede estar holgado debido a una carencia temporal de trabajo. La manera más común de evitar tal tiempo de ocio es proteger el recurso del cuello de botella con un buffer de trabajo. El buffer tiene la intención de absorber la variabilidad en la tasa de llegada del nuevo trabajo mediante una cola; en este ejemplo para nuestro diseñador de experiencia de usuario. Generar buffers agrega WIP total a nuestro sistema. Desde una perspectiva Lean agregar un buffer al trabajo añade desperdicio e incrementa el tiempo de espera. Sin embargo, la ventaja en throughput que se da asegurando un flujo estable de trabajo a través de nuestro recurso de capacidad restringida es usualmente un buen trato. Obtendrá mayor trabajo hecho a pesar de tener un tiempo de entrega un poco más largo y un total de trabajo-en-progreso un poco más grande.

Utilizar buffers para asegurar que un recurso del cuello de botella esté protegido de tiempo ocioso es conocido como “protegiendo al cuello de botella” o como una acción de protección. Antes de considerar elevar un cuello de botella debe buscar maximizar la explotación y la protección para asegurar que la capacidad disponible es utilizada lo más posible.

Nuestro ejemplo de administración de tráfico del SR-520, donde el throughput actual era menor al 20% de su potencial resulta ser bastante común en problemas de trabajo del conocimiento tales como un análisis de requerimientos y desarrollo de software. Es a veces posible ver mejoras de hasta cuatro veces en la tasa de entrega con tal solo explotar un cuello de botella.

Acciones de subordinación

Una vez que ha decidido cómo explotar y proteger al recurso de capacidad restringida puede necesitar tomar acciones – subordinar otras cosas en el sistema – para hacer su esquema de explotación funcionar efectivamente.

Visitemos nuevamente nuestro sistema de tráfico de fantasía en un mundo futuro. Aquí decidimos no construir un nuevo puente flotante que atraviesa el lago Washington; en lugar de ello, decidimos instalar en todos los vehículos que viajan en el SR-520 en horas pico un nuevo sistema de control de velocidad que utiliza radar y comunicación inalámbrica para regular la velocidad del tráfico en un tramo de siete millas de la carretera. Este nuevo sistema actuará como un sistema de crucero y anulará el uso manual de los pedales de acelerador y freno. A los ciudadanos se les hará una rebaja en sus impuestos como incentivo. Una vez que existan suficientes vehículos con el sistema éste será activado y los vehículos que no lo tengan tendrán que encontrar una ruta alterna o cruzar fuera de las horas pico. El resultado va a ser tráfico de flujo más uniforme y mayor capacidad de explotación en el cuello de botella. Yo supondría que si tal sistema pudiera ser efectivo, ganaría como el 50% de la capacidad perdida. Puesto de otra manera, incrementaría el throughput a través de la SR-520 durante las horas pico alrededor de 2.5 veces.

¿Qué hemos hecho con este ejemplo? Hemos subordinado el derecho del conductor de afectar y controlar su propia velocidad buscando una meta común mayor de tiempos de viaje más rápidos mediante un mayor throughput a lo largo del puente. Esta es la esencia de una acción de subordinación. Algo más necesitará cambiarse con el fin de mejorar la explotación en el cuello de botella.

Para aquellos con conocimiento de la Teoría de Restricciones es en ocasiones contra intuitivo darse cuenta que los cambios requeridos para mejorar el rendimiento en un cuello de botella usualmente no se hacen en el cuello de botella. Mientras revisaba el manuscrito de mi primer libro²⁴, un miembro de la comunidad de desarrollo ágil de software muy conocido sugirió que utilizara la Teoría de Restricciones como un enfoque de mejora que llevaría a que todos en el equipo desearan ser parte del recurso de cuello de botella porque obtendrían toda la atención de la dirección. Este es un error fácil de cometer. Contraintuitivamente, la mayoría de la administración del cuello de botella sucede fuera del cuello de botella. Muchos de los cambios se enfocan en reducir la carga de falla hacia el cuello a fin de maximizar su throughput. Como regla general espere maximizar la explotación de la capacidad del cuello de botella y por lo tanto maximizar throughput; y como resultado, minimizar el tiempo de entrega en su proyecto tomando acciones sobre toda la cadena de valor y muy probablemente no en el cuello de botella mismo.

Recursos de disponibilidad no-instantánea

Los recursos de disponibilidad no-instantánea no son, estrictamente hablando, cuellos de botella, sin embargo, se parecen mucho y las acciones que podríamos tomar para compensarlos son similares en naturaleza a aquellas para los cuellos de botella. Cualquiera que ha llegado a conducir un auto y se ha detenido en un semáforo entiende el concepto de disponibilidad no instantánea. Mientras que esté detenido en la luz roja, el auto no puede fluir a lo largo del camino. La carencia de flujo no es causada por restricciones de capacidad en el camino sino por una política que permite que los vehículos en el otro camino tengan el derecho de cruzar el camino en el que nosotros estamos.

Un mejor ejemplo y ajustándonos al tema en este capítulo sobre transporte en el estado de Washington, sería el sistema de transbordador que opera a través de Puget Sound uniendo la península Kitsap y la península Olímpica con el continente alrededor de la ciudad de Seattle. Hay tres rutas de transbordador, dos que salen de Seattle cruzando Bremerton y Bainbridge y mi favorito, la SR-104 que cruza Edmonds en el lado Este hacia Kingston en el oeste. En un mapa, la ruta del transbordador se muestra como parte de la ruta SR-104. Está normalmente marcado como “caseta de cobro”, en lugar de indicar explícitamente “tiene que subirse al barco aquí ;-). La gente de transporte considera el transbordador como un camino de disponibilidad no instantánea.

Cuando usted llega al muelle del transbordador paga y se le pide que permanezca en el área de espera. El tiempo de espera habitual es de alrededor de 30 minutos que es lo que aproximadamente tarda el transbordador en cruzar Puget Sound y hay un período de 10 a 15 minutos para descargar los vehículos y otro similar para cargar los nuevos vehículos antes de zarpar. Usualmente la compañía del transbordador opera dos barcos los cuales zarpan aproximadamente cada 50 minutos. En las horas pico pueden operar tres barcos en la ruta, acortando el tiempo de espera alrededor de 35 minutos.

La mayoría del tiempo el transbordador zarpa casi lleno pero el sistema no está restringido por capacidad. El hecho de que los vehículos se unan a la cola en el área de espera –un buffer—y sean cargados en el transbordador (transferencia de lote) no indica un recurso de capacidad restringida. Sin embargo, indica un recurso de disponibilidad no instantánea. Los transbordadores zarpan una o dos veces por hora con una capacidad de alrededor de 220 vehículos cada vez.

Durante las horas pico, tales como la tarde de los viernes, el sistema de transbordador se hace de capacidad restringida. Cuando esto sucede, la tasa de llegada de vehículos deseando cruzar excede la capacidad de transportarlos. La capacidad es alrededor de 300 vehículos por hora. Los vehículos se acumulan, son puestos en cola fuera del área de espera antes de la caseta de cobro. Durante esos tiempos de pico de demanda, a menudo puede ver vehículos alineados a lo largo de 2 millas a través de Edmonds o Kingston. Muy poco se puede hacer y los vehículos simplemente tienen que esperar. No es fácil elevar la restricción trayendo otro transbordador. El itinerario de los transbordadores está diseñado para proveer un nivel razonable de servicio en un tiempo razonable. Poder tener siempre capacidad en exceso sería muy costoso o se tendrían que pagar muchos impuestos para subsidiar el servicio de transbordador.

Regresando a desarrollo de software y a trabajo del conocimiento, la disponibilidad no-instantánea tiende a ser un problema con recursos compartidos o gente a quien se le pide que hagan mucha multitarea. Como todos sabemos, en realidad no hay tal cosa como multitarea en la oficina; lo que en realidad hacemos es cambio frecuente de tareas. Si nos piden que trabajemos en tres cosas simultáneamente, trabajamos en la primer cosa por un rato, luego cambiamos a la segunda y luego a la tercera. Si alguien está esperando a que terminemos la primer cosa mientras que trabajamos en la segunda o la tercera, aparentaríamos estar no-instantáneamente disponibles desde la perspectiva de esa persona (y de esa tarea).

Un ejemplo de disponibilidad no-instantánea lo observé con un ingeniero de build. La compañía tenía una política de que se solamente el personal del equipo de gestión de la configuración tenía permitido construir código y empujarlo al entorno de pruebas. Esta política era una estrategia específica de administración de riesgos basada en experiencia histórica de que los desarrolladores eran con frecuencia descuidados y construirían código que rompería el entorno de pruebas. El entorno de pruebas era compartido frecuentemente entre varios proyectos, por lo que el impacto de una versión del software con errores era

significativo. El departamento de tecnología no tenía una buena capacidad de coordinación a nivel de programa y la posibilidad de que un equipo y un proyecto estuvieran trabajando en un área de los sistemas agregados de TI que podría afectar al de otro proyecto era muy alta. La función de coordinación de saber qué estaba pasando a nivel técnico de código entre, y a través de proyectos se asignaba al departamento de gestión de la configuración. A estos profesionales se les conocía como ingenieros de build. El ingeniero de build era responsable de saber el impacto de un conjunto de cambios en una versión del software dado y de evitar romper el sistema de pruebas tal que el flujo de todos los proyectos no fuese afectado por una falla en el entorno de pruebas.

Un proyecto generalmente tenía asignado a un ingeniero del equipo compartido de recursos de gestión de configuración. Sin embargo, la demanda por builds de código de un solo proyecto en pruebas no era suficiente para mantener a un ingeniero de build ocupado todo el día. De hecho, en general no le mantenía ocupado por más de una o dos horas cada día. Por lo tanto, se les pedía a los ingenieros de build que efectuaran varias tareas simultáneamente. Se les asignaba o bien a varios proyectos, o bien a otras labores.

Tome el ejemplo de Doug Burros de Corbis; había sido asignado como el ingeniero de build para la actividad sustentable de ingeniería. También le habían asignado otras dos labores: tenía responsabilidades para construir nuevos entornos de desarrollo y de mantener ambientes actuales. El era el ingeniero de gestión de configuración con responsabilidad total de mantener la configuración actualizada. Esto incluía aplicar parches y actualizaciones al sistema operativo y al servidor de base de datos, parches y actualizaciones al middleware, configuración del sistema y topología de redes, etc. El dedicaba alrededor de una hora diaria a la labor de ingeniería de build, habitualmente por la mañana, de las 10:00 a las 11:00. Si los desarrolladores requerían un build para pruebas a las 3:00 PM, normalmente esperarían hasta el siguiente día. El ingeniero de build no estaba disponible instantáneamente. El trabajo se bloqueaba y, como la ingeniería de sostenimiento operaba usando un sistema kanban, el trabajo era puesto en cola a lo largo de toda la cadena de valor, causando tiempo de ocio para otros miembros del equipo.

Las acciones tomadas en respuesta a los problemas de disponibilidad no-instantánea con el flujo son remarcablemente similares a aquellas para un recurso de capacidad restringida.

Acciones de explotación/protección

Lo primero era reconocer que Doug era un recurso de disponibilidad no-instantánea y observar el impacto que esto estaba teniendo. El trabajo se estaba en cola cuando él no estaba disponible porque los límites de kanban estaban bien definidos. Debido a que Doug era una fuente de variabilidad en el flujo, el curso de acción correcto era poner un buffer de trabajo frente a Doug. El truco fue hacer su buffer lo suficientemente grande para permitir que el flujo continuara sin hacerlo demasiado grande al punto tal que Doug se convirtiera

en un recurso de capacidad restringida. Tuve una discusión con él sobre la naturaleza de la actividad de builds. Resultó que él podría hacer un build de hasta un total de siete ítems dentro de su hora de disponibilidad diaria, por lo que creamos un buffer con límite de kanban de siete. Introducimos esto a la cadena de valor y la pared de tarjetas agregando una nueva columna llamada *Lista para Builds*. De hecho incrementamos el WIP total potencial de todo el sistema en un 20%, pero funcionó. Aunque los builds no estaban disponibles instantáneamente, las actividades en los niveles anteriores podían continuar fluyendo durante el día. El resultado fue un impulso significativo en el throughput y tiempos de entrega más cortos a pesar del incremento del WIP. Otra opción que, de nuevo, no pensamos en ese entonces habría sido preguntarle a Doug que trabajara dos turnos de media hora en lugar de un turno de una hora. Estos dos tiempos estarían divididos durante el día: en la mañana y en la tarde. Esto habría aliviado el flujo. El efecto habría sido reducir la presión de poner en buffer la disponibilidad no-instantánea. Tal vez el tamaño del buffer se podría haber reducido a dos o tres. Esto habría tenido un impacto de tan solo el 10% del WIP total y habría resultado en tiempos de entrega más cortos a través del sistema.

Como regla general, al encontrar problemas de disponibilidad no-instantánea piense como mejorar la disponibilidad. Lo último es tornar un problema de disponibilidad no-instantánea en un recurso disponible instantáneamente.

Acciones de subordinación

Como se discutió anteriormente, las acciones de subordinación generalmente involucran hacer cambios de política a través de la cadena de valor para maximizar la explotación del cuello de botella. ¿Qué opciones están disponibles como acciones de subordinación con nuestro ingeniero de build disponible no-instantáneamente?

Lo primero fue examinar la política que le pedía a Doug que efectuara tres funciones distintas. ¿Era esa la mejor elección? Discutí esto con su director. Parecía ser que en su equipo, los ingenieros querían y necesitaban diversidad de trabajo para mantenerlo interesante. También, pidiéndoles a los miembros del equipo que efectuaran builds del sistema, mantenimiento del sistema, e ingeniería de builds, se mantenía el conjunto de habilidades generales a través de todos los miembros del equipo manteniendo el conjunto de recursos flexible. Esto le daba al director muchas más opciones y evitaba el potencial de cuellos de botella de recursos de capacidad restringida debida al alto grado de especialización. La generalización era atractiva también para los miembros del equipo desde el punto de vista de carrera y curriculum. No deseaban hacerse demasiado limitados en habilidades. Por lo que pedirle al equipo que trabajaran en una sola área tal como ingeniería de builds no era deseable.

Otra opción habría sido abandonar la idea de multitareas y dedicar Doug al equipo de ingeniería sustentable. Esto le habría provisto de mucho tiempo holgado. Estaría sentado

esperando por trabajo, como un bombero en su estación esperando por una llamada para apagar un fuego. Manteniendo a Doug esperando habría aliviado el problema de flujo pero era esa una elección razonable?

Los presupuestos eran limitados y agregar gente al equipo de administración de configuración para encargarse del build y mantenimiento del sistema y que Doug estaba haciendo sería caro y tal vez imposible. Necesitaría pedirle a mi jefe presupuesto para obtener otra persona porque quería mantener a alguien ocioso la mayoría del tiempo. ¿Era este un buen compromiso de administración del riesgo?

Para decidir esto necesitamos ver el costo de retraso para el esfuerzo de ingeniería de mantenimiento y comparar el costo de otro miembro del personal con el costo de otras alternativas para mantener flujo. La realidad era que muy pocos ítems en la cola de ingeniería sustentable tenían un costo de retraso estratégicamente significativo. Por lo que la idea de que tuviéramos a alguien holgado esperando por trabajo con el fin de optimizar flujo no era una alternativa viable. Claramente, la acción de explotación de agregar un buffer de trabajo para mantener flujo era una alternativa mejor y más barata.

Sin embargo, la discusión sobre qué hacer respecto a la carencia de disponibilidad instantánea de Doug creó un debate en el equipo acerca de la política de tener haciendo sólo esta labor a los ingenieros de build. Discutimos la opción de terminar la política y permitir a los desarrolladores que construyeran y empujaran código en el ambiente de trabajo. Fue rechazado porque la organización no tenía un método alternativo viable para coordinar los riesgos técnicos a través de los proyectos. La opción de proveer un entorno de pruebas dedicado para ese proyecto fue rechazado por motivos de costo y no era una alternativa práctica viable en el corto o mediano plazo. Todos continuaron viendo el valor en la función del ingeniero de build y el equipo de gestión de configuración.

Acciones de elevación

Sin embargo, generar buffers y agregar WIP para resolver el problema se sintió como una banda médica sobre la herida. Se sentía como una “solución” que le daba la vuelta al asunto. Y podría verlo de esa manera. Era una solución táctica—una solución táctica efectiva, pero una solución táctica a fin de cuentas—que acarreaba un costo. Porque el sistema kanban había expuesto el cuello de botella de disponibilidad no instantánea y permitía que el equipo tuviera un debate completo sobre su causa y las soluciones posibles, la discusión inevitablemente llegó a la pregunta de si tener a un ser humano haciendo los builds del código era la respuesta correcta. ¿Sería posible automatizar este proceso de build? La respuesta era “Sí” aunque la inversión era grande. Se necesitaría un desarrollo considerable de capacidad en administración de la configuración y coordinación entre proyectos. Adicionalmente, algún especialista en automatización necesitaría ser contratado por un período de tiempo para crear el sistema y hacerlo funcionar.

Tomó alrededor de seis meses de tiempo total y dos contratistas durante ocho semanas. El costo financiero total fue de alrededor de \$60,000. Sin embargo, el resultado final fue que Doug ya no era requerido para hacer builds y los builds eran instantáneamente disponibles cuando los desarrolladores los necesitaban. En ese punto, fue posible eliminar el buffer y reducir el WIP del sistema. Lo cual a su vez resultó en una reducción del tiempo de espera.

La automatización fue ultimadamente la ruta para la elevación del cuello de botella de disponibilidad no-instantánea. Aumentando la capacidad, es decir, la contratación de otro ingeniero, no era una buena elección.

Otra ruta que involucraba la automatización también fue tomada—virtualización de entornos de desarrollo. La virtualización ya era común en nuestra industria; sin embargo, en ese entonces nuestros ambientes aún eran físicos. La virtualización no era una capacidad organizacional. Tomando el tiempo de hacerlo, el entorno de pruebas podría ser configurado y restaurado fácilmente. Esto reduciría el impacto de que un build rompiera el ambiente; en la gestión de riesgos ésta es una estrategia de mitigación. También permitiría entornos dedicados—reduciendo o eliminando el riesgo de que un build rompiera la configuración de otro proyecto.

Por lo que generar buffers se utilizó como una estrategia de explotación táctica de corto plazo mientras que la automatización se perseguía como una estrategia de elevación de largo plazo.

¿Y que hay en cuanto a nuestro ejemplo del transbordador de Edmonds a Kingston? Bueno pues el Estado de Washington está considerando dos opciones actualmente. Una sería reemplazar la flota actual de transbordadores viejos por un nuevo conjunto de botes más grandes y eficientes. Sin embargo, Washington tiene mucha experiencia con puentes flotantes. Hay dos a través del Lago Washington, incluyendo el de la SR-520, el cual es aparentemente el más largo del mundo en su tipo y otro a través del Canal Hood en SR-104. Lo que ahora está bajo consideración es la posibilidad de construir un nuevo puente flotante que rompa el récord a través de Puget Sound como parte de SR-104 y reemplazar el servicio de transbordador enteramente. El puente planificado no tan solo resolverá el problema de restricción de capacidad durante las horas pico, también resolverá los problemas de disponibilidad no-instantánea que obstaculizan todos los servicios de transbordador como una opción de control de tráfico. Tal puente abriría las penínsulas Kitsap y Olímpica para un crecimiento económico más rápido. ¿Probablemente en 50 años alguien más estará escribiendo un libro discutiendo como el puente flotante de la SR-104 a través de Puget Sound es un cuello de botella y un recurso de capacidad restringida durante las horas pico?

Para llevar

- ❖ A menudo, los cuellos de botella en el proceso están rindiendo muy por debajo de su capacidad potencial—bajo la restricción de capacidad teórica.
- ❖ El throughput en el cuello de botella puede ser mejorado hasta el límite de la restricción de capacidad teórica utilizando acciones de explotación y protección.
- ❖ La protección habitualmente involucra agregar un buffer de WIP frente al cuello de botella. Esto es cierto para recursos de capacidad restringida o recursos de disponibilidad no-instantánea.
- ❖ Las acciones de explotación habitualmente involucran cambios de políticas que controlan el trabajo hecho por el recurso de cuello de botella.
- ❖ Las clases de servicio pueden usarse como acciones de explotación.
- ❖ Las acciones de subordinación son acciones que se toman en cualquier parte de la cadena de valor para permitir las acciones de explotación o protección deseada.
- ❖ Las acciones de explotación, protección y subordinación son con frecuencia fáciles y baratas de implementar debido a que involucran principalmente cambios de política. Por ende, maximizando el throughput de un cuello de botella mediante su explotación total se puede ver como una mejora táctica de proceso.
- ❖ A pesar de la naturaleza táctica de explotación de un cuello de botella, las ganancias son a menudo dramáticas con mejoras de entre dos y media a cinco veces en throughput, con caídas consecuentes de tiempo de entrega, pueden ser alcanzadas a menudo bajo un costo mínimo o nulo sobre un período corto de meses.
- ❖ La explotación debe ser perseguida primero en todos los casos antes de intentar elevación
- ❖ No es inusual que un conjunto táctico de acciones de explotación y subordinación sean implementados mientras que un plan de cambio estratégico para elevar una restricción es implementada en un período largo de tiempo.

❖ CAPÍTULO 18 ❖

Un modelo económico para lean

Desperdicio (en japonés: *muda*) es la metáfora utilizada en Lean (y el Sistema de Producción Toyota) para actividades que no agregan valor al producto final. La metáfora de desperdicio ha sido problemática con los trabajadores de conocimiento. Con frecuencia es difícil aceptar como desperdicio tareas o actividades que son costos o aspectos extra pero que son necesarios o esenciales para completar trabajo de valor agregado; por ejemplo, las reuniones de pie diarias son esenciales para coordinar a la mayoría de los equipos. Estas reuniones no agregan valor directamente al producto final, por lo que técnicamente son “desperdicio”, pero esto ha sido difícil de aceptar por muchos practicantes del desarrollo ágil. En lugar de tener a las personas argumentando obsesivamente sobre qué es y qué no es un desperdicio he concluido que sería mejor encontrar un paradigma alternativo y un lenguaje alternativo que es menos confuso o emocionalmente evocativo.

Redefiniendo “desperdicio”

Siguiendo la guía de escritores tales como Donald G. Reinertsen, he adoptado el uso del lenguaje de economía y me refiero a estas actividades de “desperdicio” como costos. Clasifico los costos en tres categorías abstractas principales: Costos de Transacción, Costos de Coordinación y Carga de Falla. La Figura 18.1 ilustra esto.

La Figura 18.1 muestra que hay un número de actividades de valor agregado para una iteración o proyecto sobre el tiempo. Alrededor de estas actividades hay costos de coordinación y transacción. La capacidad para las actividades de valor agregado puede ser desplazada por trabajo que puede ser considerado la carga de falla; es decir, trabajo que en realidad es re-hecho o demanda puestos en el sistema debido a una pobre implementación previa.



Figura 18.1 El Modelo de Costo Económico para el Desarrollo de Software Lean

Cada uno de estos costos está discutido en detalle en las siguientes secciones. Los describiré utilizando un ejemplo simple de la vida real – la actividad de pintar la cerca de mi casa en Seattle con un preservador de madera.

Costos de transacción

La cerca tiene 21 secciones. El valor al cliente es entregado cuando una sección de la cerca es pintada con el preservador. El valor completo es entregado cuando el total de las 21 secciones han sido pintadas por ambos lados. Antes de que pudiera comenzar el trabajo, primero tuve que adquirir todos los materiales. Esto involucró un viaje a Home Depot. También se requirió un trabajo de preparación de la cerca: algunas reparaciones como lijado y cortar plantas y arbustos para permitir el acceso. Ninguna de estas actividades se podría describir como de valor agregado. Al cliente no le importa que tenga que hacer un viaje a Home Depot, tampoco le importa que esta actividad lleve tiempo. De hecho, es molesto

porque retrasa el inicio y el fin del proyecto. Todas estas actividades retrasan la entrega de valor al cliente.

Es decir, el proyecto tiene algunas actividades de preparación que son esenciales antes de que el trabajo de valor agregado pueda comenzar.

Puede haber otras. Puede haber alguna planificación o alguna actividad de estimación y algún establecimiento de expectativas. Se le pudo haber dado al cliente un precio por el trabajo y una fecha de entrega. (En este caso, el cliente del proyecto fue mi esposa).

En cuanto al hecho de pintar la madera, resulta que 42 secciones de cerca son demasiadas para intentarlas en una sola sección. La velocidad de pintado fue de aproximadamente cuatro secciones por hora. Por lo que el trabajo se dividió en seis segmentos. Si esto fuera desarrollo de software, les podríamos haber llamado iteraciones o sprints. Si fuera manufacturación, les podríamos haber llamado lotes. Cuando iba a comenzar a pintar un segmento, también tuve algunas actividades de preparación. El primero fue cambiarme de ropa, luego preparar los materiales: cargar la pintura, las brochas y otras herramientas del garaje a la ubicación de lo que pintaría ese día, sólo entonces comenzaría a pintar.



En resumen, tanto proyectos como iteraciones tienen actividades de preparación.

Después de un par de horas de pintado, puede que desee tomar un descanso. Probablemente es hora del almuerzo. Puedo dejarlo todo y comenzar a comer. Primero debo asegurar la pintura poniendo la tapa en la lata y después debo asegurar las brochas limpiándolas o poniéndolas en una jarra de agua para prevenir que se endurezcan mientras tomo el descanso. Después me limpio. Me lavo las manos y me cambio la ropa de trabajo; y entonces ya puedo ir a comer.

Cuando todo el proyecto se completa, puede que tenga alguna pintura extra y las latas llenas se pueden ser regresadas a Home Depot para obtener un reembolso. Por lo que otro viaje es requerido.

Parece que tanto las iteraciones como los proyectos tienen un conjunto de actividades de limpieza.

En términos económicos estas actividades de preparación y limpieza son referidas como costos de transacción. Estas actividades de costos de transacción son cosas que el cliente puede no ver y probablemente no valora y hacia las cuales son ambivalentes a lo mucho. El cliente puede ser forzado a pagar los costos de estas actividades pero preferiría no hacerlo. ¿Con qué frecuencia ha llamado a un plomero para arreglar una máquina de lavado o una

lavadora de platos y le ha pedido una cuota de llamada de \$90? Esto es un costo de transacción. ¿Preferiría una tarifa más baja? ¿Elegiría a un plomero que no le cargue tal tarifa? Los costos de transacción no agregan valor. Puede que sean necesarios, pero en términos Lean, son un desperdicio.

Por lo que los primeros dos tipos de desperdicio son costos de transacción, específicamente: los costos de transacción de la preparación o front-end y de la limpieza o back-end.

Si usted considera esto en actividades de desarrollo de software se dará cuenta de que todos los proyectos tienen un número de actividades de preparación, tales como la planificación de proyectos, la planificación de la comunicación, la adquisición de instalaciones, etc. La mayoría de los proyectos también tiene costos de limpieza y otros costos de transacción relacionados de back-end, tales como la entrega al cliente, desmantelar los entornos, retrospectivas, revisiones, auditorías, capacitación a usuarios, etc.

Las iteraciones también tienen costos de transacción, incluyendo la planificación de iteración y la selección del *backlog* (o el alcance de requerimientos), tal vez la estimación, asignación de presupuesto, asignación de recursos y preparación de entornos. En el back-end tendrían costos de transacción que incluyen integración, entrega, retrospectiva y desmantelamiento de entorno.

Costos de coordinación

La coordinación es necesaria tan pronto como dos o más personas intentan lograr una meta común juntos. Inventamos sistemas de lenguaje y comunicaciones para coordinarnos entre los seres humanos. Cuando acordamos reunirnos con los amigos, tomar bebidas, cenar y ver una película en la noche de un viernes, incurrimos en gastos de coordinación. Todos los correos electrónicos, mensajes de texto y llamadas telefónicas que son requeridas para preparar una noche social son los gastos de coordinación.

Los gastos de coordinación en proyectos son cualquier actividad que involucra comunicación y programación. Cuando la gente de los equipos de trabajo se quejan porque no pueden hacer ningún trabajo que agrega valor—e.g., el análisis, el desarrollo, o las pruebas—porque están enviando correos electrónicos, están llevando a cabo un conjunto de actividades de coordinación: cada correo electrónico leído y contestado es una actividad de coordinación. Cuando se quejan de que no pueden hacer trabajo que agrega valor—e.g., el análisis, la programación, o las pruebas—porque siempre están en reuniones, esas reuniones también son actividades de coordinación.

En general, cualquier forma de reunión es una actividad de coordinación, incluyendo las favoritas dentro de la comunidad ágil, tales como la reunión de pie diaria, a menos que la reunión esté diseñada para producir un entregable valorado por el cliente. Si tres desarrolladores se sientan juntos frente a un pizarrón blanco y modelan el diseño para el código que está por implementar, esa no es una actividad de coordinación, es una actividad

de valor agregado. ¿Por qué? Es de valor agregado porque produce información que contribuye a una función completa valorada por el cliente.

Si vemos el desarrollo de software y sistemas como un proceso de llegada de información en el cual nuestro punto de inicio es no información, e información completa representa funcionalidad que funciona y satisface las necesidades e intención del usuario, entonces cualquier información que llega entre los puntos de entrada y fin—que nos mueve adelante hacia la funcionalidad que trabaja y satisface las necesidades del cliente—es información de valor agregado.

Si los miembros del equipo se reúnen con el fin de crear información de valor agregado, tal como un diseño, una prueba, un pedazo de análisis, o una sección de código, entonces esa reunión no es un costo de coordinación, es una actividad de valor agregado.

Sin embargo, si los miembros del equipo se reúnen para discutir el estado, asignación de tareas, o programar fechas que les ayudan a coordinar sus acciones y el flujo de entregables, esa reunión es un costo de coordinación y debe considerarse como un desperdicio. Como tal, usted debe buscar cómo reducir o eliminar las reuniones de coordinación.

Por lo tanto, una reunión de pie de 5 minutos es mejor que una reunión de pie de 15 minutos si logra el mismo monto de coordinación. Y una reunión de pie de 15 minutos es mejore que una reunión de pie de 30 minutos si logra el mismo monto de coordinación.

Puede pensar en la reducción de actividades de coordinación encontrando otras maneras mejores de coordinar gente.

Una manera es facultar a los miembros del equipo para que se auto-organicen. Dirección basada en comando-y-control , en la cual la gente se reúne para asignar tareas a los individuos por adelantado es un desperdicio. Es mejor dejar que los miembros del equipo se auto-asignen tareas. La auto-organización generalmente reduce los costos de coordinación en un proyecto. Sin embargo, requiere información para poder funcionar.Técnicas dentro de Kanban—tales como el uso de seguimiento visual de la cadena de valor, visualización de trabajo utilizando una pared de tarjetas y herramientas y reportes electrónicos—proveen información de coordinación que permiten auto-organización y reducen los costos de coordinación en un proyecto. El uso de las clases de servicio y su visualización con tarjetas de colores o carriles de nado en la pared de tarjetas, junto con el conjunto asociado de políticas para la clase de servicio, permiten auto-organización para programación de itinerario y priorización automática. A esto se le refiere como “auto-expansible” (un término que asocio con Eli Goldratt en referencia a la administración de buffers).

En general, cuanta mayor información se pueda hacer transparente para los trabajadores de conocimiento en el equipo, mayor auto-organización será posible y menores actividades de coordinación serán requeridas. Permita que la transparencia de trabajo, el flujo de proceso y las políticas relacionadas a la administración de riesgos desplacen a las actividades de coordinación. Reduzca el desperdicio a través de un uso más amplio de transparencia.

¿Cómo sabe si una actividad es un costo?

He descubierto que mucha gente tiene problemas identificando actividades que son desperdicio. He visto, por ejemplo, a defensores de ágil discutir que las reuniones de pie diarias son de valor agregado. Yo no me suscribo a ese punto de vista. No puedo medir como a un cliente le importa si un equipo lleva a cabo reuniones de pie o no. Lo que el cliente desea es funcionalidad que permita que sus metas sean entregadas a tiempo y con alta calidad. Si un equipo necesita o no llevar a cabo reuniones de pie diarias para permitir tal entrega no está ni aquí ni allá desde su perspectiva.

¿Cómo identifica entonces los costos de transacción o actividades de coordinación que son desperdicio?

Creo que se debe preguntar, “¿si esta actividad esta realmente agregando valor, haríamos más de ella?”

Cuando se le pregunta a los defensores de Scrum quienes argumentan con vehemencia que las reuniones de pie diarias son de valor agregado, si ellos la llevarían a cabo dos veces al día o si las harían más largas, de 15 a 30 minutos, con certeza contestarán, ”¡No!”

“Bueno, si las reuniones de pie realmente son de valor agregado”, les contesto, “¡entonces seguramente hacer más sería algo bueno!”

Esta es la prueba de fuego real que demuestra la diferencia entre una actividad que realmente agrega valor y un costo de transacción o coordinación. Desarrollar más requerimientos del cliente es claramente una agregación de valor. Usted haría más de eso si pudiera y el cliente pagaría con gusto por ellas. La planificación no es de valor agregado claramente. El cliente no pagaría por más planificación si pudiera evitarlo.

Pregúntese, “¿haríamos más de esto?” Rete a otros con la misma pregunta sobre actividades que realizan. Si la respuesta es “No” entonces considere como minimizar el tiempo y energía gastadas en esa actividad, o como podría hacer la actividad más efectiva y entonces reducir la duración, frecuencia, o cantidad de la actividad.

A veces puede ser difícil determinar si una actividad es un costo de transacción o un costo de coordinación. Algunas actividades a menudo parecen ser ambas. Veo esta confusión todo el tiempo cuando enseño mi clase de Kanban. Así como lo hago con los participantes de la clase, le sugiero que no gaste mucha energía intentando determinar la diferencia. Lo que es importante es que ha identificado una actividad como que no agrega valor y es por lo tanto un desperdicio—y usted sabe que desea reducir o eliminar esa actividad como parte de un programa de mejora continua.

Carga de falla

La carga de falla es la demanda generada por el cliente que podría haber sido evitada mediante mayor calidad entregada inicialmente. Por ejemplo, muchas llamadas al servicio

de asistencia generan costos para el negocio. Si el servicio o producto de software o tecnología fuera de mayor calidad, más usable, más intuitivo, más adecuado para el propósito, entonces habría menos llamadas. Esto le permitiría al negocio reducir el monto de personal del centro de llamadas y reducir costos.

Muchas llamadas de ayuda tienden a generar muchos tickets de defectos en producción. Al seleccionar las funciones en el alcance para un proyecto o iteración, el negocio debe escoger entre nuevas ideas y defectos de producción. Los defectos de producción no son tan solo defectos de software; incluyen problemas de uso y otros problemas no funcionales tales como pobre ejecución, falta de respuesta bajo carga o ciertas condiciones de red, etc. La solución para un defecto en producción contra un requerimiento no funcional puede parecer ser nueva funcionalidad—el diseño de una nueva pantalla probablemente—pero en realidad no lo es. Es una carga de falla. Ese nuevo diseño de pantalla surgió por un defecto de uso en una entrega anterior.

La carga de falla no genera nuevo valor; habilita el valor “dejado en la mesa” de una entrega anterior. Muy probablemente la entrega inicial del producto o servicio falló en la entrega de su función de pago proyectada. Aunque algo de esto puede ser debido a variabilidad de mercado o la imprevisibilidad, algo del déficit se habría descubierto que viene de problemas con una entrega anterior. Tal vez un defecto en el producto previene del uso de alguna funcionalidad. Debido a esto, clientes potenciales se alejan del producto y difieren su adquisición o eligen un producto rival.

El producto está entonces sucio. La carga de falla aún agrega valor. Pero lo que es importante es que agrega valor que ya debería haberse tenido. Reduciendo la carga de falla se reduce la oportunidad de costo de retraso. Costos reducidos significan más ganancias más pronto. La reducción de carga de falla significa que más de la capacidad disponible puede ser utilizada en nueva funcionalidad. La carga de falla reducida le permite al negocio buscar más nichos de mercado, con mayor oferta de producto. La carga de falla reducida permite más opciones. La carga de falla reducida puede permitir una reducción en el tamaño del equipo y por tanto, una reducción en costos directos.

Para llevar

- ❖ El desperdicio se puede clasificar en tres categorías abstractas principales: costos de transacción, costos de coordinación y carga de falla.
- ❖ El concepto de desperdicio es una metáfora.
- ❖ La metáfora de desperdicio no funciona bien para todos, pues el desperdicio es a veces necesario aunque no agrega valor específicamente. Como resultado, lo he reemplazado por un modelo de costo económico.
- ❖ Para determinar si una actividad es realmente un desperdicio pregunte, “¿Haríamos más de esto si pudiéramos?” Si la respuesta es no, entonces la actividad es alguna forma de desperdicio.
- ❖ Los costos de transacción vienen en dos tipos: actividades de preparación y de limpieza, o actividades de entrega.
- ❖ Los costos de coordinación son actividades efectuadas con el fin de asignar tareas a personas, programar el horario de eventos, o coordinar el trabajo de dos o más personas hacia un resultado común.
- ❖ La carga de falla es nuevo trabajo de valor agregado que es generado debido a una falla previa, tal como un defecto en el software, o un diseño o implementación pobre que llevó a una carencia de adopción del cliente, una falla de realización de una función de pago de objetivo, o un incremento significativo en llamadas al centro de ayuda o solicitudes de servicio.
- ❖ La carga de falla utiliza capacidad que podría ser usada en características nuevas, innovador, de valor adicional al cliente y características generadoras de ingresos.
- ❖ Convertir ideas en código funcional y entregable al cliente más rápidamente maximiza el valor potencial y minimiza el desperdicio.

❖ CAPÍTULO 19 ❖

Fuentes de variabilidad

La variabilidad y los procesos industriales han sido estudiados desde los inicios de la década de los veintes. El pionero en este tema fue Walter Shewhart. Sus técnicas se convirtieron en el fundamento del movimiento de control de calidad (quality assurance) y los elementos fundamentales tanto del Sistema de Producción Toyota y los métodos de Seis Sigma de calidad y mejora continua. Las técnicas de Shewhart fueron adoptadas, desarrolladas y avanzadas por W. Edwards Deming, Joseph Juran y David Chambers. Su trabajo fue inspirador para Watts Humphrey y los miembros fundadores del Software Engineering Institute en la Universidad de Carnegie Mellon, quien mantuvo la creencia de que el estudio de la variación y su reducción sistemática traerían grandes beneficios a la profesión de la ingeniería de software.

Hay mucho material publicado por Shewhart, Deming y Juran sobre el estudio de la variación y su uso como una técnica de gestión y el fundamento para un programa de mejoras. Adicionalmente, se ha publicado mucho sobre el método de evaluación cuantitativa conocido como Control Estadístico de Proceso (en inglés: *Statistical Process Control—SPC*) que surgió como la herramienta principal para estudiar la variación y actuar sobre ella. En el momento de escribir estas líneas, el uso de SPC está surgiendo en equipos que adoptan Kanban. Sin embargo, el uso de SPC se considera avanzado y es un tópico de alta madurez que será tratado en otro libro. Aquí hablaremos sobre variación en los términos más generales en su forma más simple.

Shewhart clasificó la variabilidad y las variaciones en el rendimiento de procesos en dos categorías: interna y externa.

Las fuentes internas de variación son variaciones que están bajo el control del sistema en operación. Con Kanban para la ingeniería de software y operaciones de TI, consideramos el sistema como un proceso que está definido por un conjunto de políticas que gobiernan la operación del sistema. Pueden estar directamente afectados por cambios hechos por individuos, el equipo, o la dirección. Cambios en las políticas cambian la operación del sistema y su rendimiento. Por lo tanto, un cambio en la definición del proceso representa un cambio que afecta las fuentes internas de variación. De manera un poco irónica, Shewhart le llamó a estas variaciones generadas internamente “variaciones de causa aleatoria” (en inglés: *chance-cause variation*). “Aleatoria” implica que la variación es al azar y que la aleatoriedad es una consecuencia directa del diseño del sistema. No implica que la aleatoriedad esté distribuida uniformemente o sigue una distribución estándar. Los cambios en el diseño del proceso por medio de cambios de política interna afectarán cualquier media, dispersión y forma de la distribución.

Usando un ejemplo general, un bateador en un juego de béisbol tiene una tasa de bateo (conocida como un promedio de bateo) que indica con qué frecuencia llega a golpear la pelota tal que el resultado es llegar a la primera base o más. Distintos bateadores exhiben diferentes tasas, con un rango habitual de entre 0.100 y 0.350. En un día cualquiera, un bateador individual puede no alcanzar su tasa típica de bateo. Esto se determina por un número de factores tales como la elección del pitcher (lanzador), que tan bien los otros jugadores golpean la pelota y los lanzamientos de pelota mismos.

Si cambiamos las reglas del béisbol para permitir, digamos, 4 strikes antes de que el bateador esté fuera, cambiaremos las probabilidades a favor del bateador en lugar del pitcher. El promedio del bateador se incrementará como resultado. Algunos de los mejores jugadores podrían llegar a alcanzar tasas de bateo que sobrepasen 0.500 como resultado de tal cambio en las reglas. Este es un ejemplo de modificación del sistema para modificar la variación de causa aleatoria dentro del sistema.

Si deseamos interpretar esto dentro de un ejemplo específico de desarrollo de software, una variación de causa aleatoria interna, sería el número de defectos creados por línea de código, por requerimiento, por tarea, o por unidad de tiempo. La media, la dispersión y la distribución de la tasa de bugs (o defectos) puede ser afectada cambiando las herramientas y procesos, tal como insistir en pruebas unitarias, integración continua y revisiones de código por colegas.

La definición del proceso utilizado por nuestro equipo, expresado como políticas, representa las reglas del juego colaborativo de desarrollo de software. Las reglas del juego determinan las fuentes y cantidad de variación interna. La ironía yace en la noción de que la variación de “causa aleatoria” de hecho está directamente bajo el control del equipo y la

administración a través de su habilidad de modificar políticas, cambiar el proceso y afectar las fuentes de variación interna.

Las fuentes externas de variación son eventos que suceden fuera del control del equipo o dirección inmediata. Son aleatoriedades que llegan de otros equipos, proveedores, clientes y “actos de Dios” aleatorios—como se les conoce en la industria de seguros; por ejemplo, una interrupción de servicio de dos semanas en una granja de servidores ocasionada por una inundación que resultó de un clima excepcionalmente húmedo y tormentoso. Las fuentes externas de variación requieren un enfoque diferente para ser administradas. No pueden ser afectadas directamente por políticas, pero se puede establecer un proceso para tratar variaciones externas efectivamente. El cuerpo de conocimiento que se relaciona directamente con ese campo es la gestión de asuntos y de riesgos.

Shewhart le llamó a las variaciones externas “variaciones de causa assignable” (en inglés, *assignable-cause variation*). Por “assignable”, implicó que alguien (o un grupo de personas) podía apuntar a la fuente del problema y describirlo consistentemente; tal como, “hubo una tormenta. Llovió muy fuerte y nuestra granja de servidores se inundó”. Las variaciones de causa assignable no pueden ser controladas por el equipo local o por la dirección, pero pueden ser predichas, se pueden hacer planes y procesos diseñados para tratarlas con cuidado.

Fuentes internas de variabilidad

El proceso de desarrollo de software y de gestión de proyectos establecido, acoplado con la madurez y capacidad organizacional de los individuos del equipo determina el número de fuentes internas de variabilidad y el grado de esa variabilidad.

Para evitar confusión, no se debe pensar sobre Kanban como un proceso del ciclo de vida del desarrollo de software o como un proceso para gestión de proyectos. Kanban es una técnica para la gestión de cambios que requiere hacer alteraciones a un proceso existente: cambios tales como agregarle límites al trabajo-en-progreso.

Tamaño del ítem de trabajo

El método de análisis utilizado para desmenuzar los requerimientos en ítems para desarrollo tiene su propio grado de variabilidad. Una dimensión de esto es el tamaño de los ítems de trabajo. La literatura inicial que describió el método de Programación Extrema explica las historias de usuario como una descripción narrativa de las características tales como un usuario final las implementaría y usaría, escritas en una tarjeta de índice. La única restricción era el tamaño de la tarjeta. El esfuerzo requerido para crear una historia de usuario era descrito como entre medio día y cinco semanas de trabajo. Al cabo de un par de años surgió una plantilla para escribir historias de usuario en la comunidad de Londres.

Como un <usuario>, quiero <característica>, para poder <entregar algún valor>

El uso de esta plantilla estandarizó grandemente la escritura de historias de usuario. Uno de los creadores de este enfoque, Tim McKinnon, me reportó en 2008 que ahora cuenta con datos para demostrar que la historia de usuario promedio era de 1.2 días de esfuerzo y la distribución de variación era entre 0.5 días y 4 días.

Éste es un ejemplo específico de reducción de variación de causa aleatoria dentro del método de Programación Extrema que se logró con tan solo pedirle al equipo que estandarizara la escritura de historias de usuario alrededor de una plantilla dada. Haciendo eso, Tim cambió las reglas del juego. Las reglas originales les pedían a los miembros del equipo que escribieran historias en tarjetas de índice de manera narrativa y las nuevas reglas les pedían que continuaran con tarjetas de índices pero que siguieran un formato de oración específico. Estos cambios están muy claramente bajo la influencia y control de directores locales. Son internos al sistema. El tamaño de una historia de usuario está controlada por la variación de causa aleatoria.

Mezcla de tipos de ítem de trabajo

Cuando todo el trabajo es tratado igual y tal vez nombrado por un mismo tipo, es probable tener mayor variación en tamaño, esfuerzo, riesgo, u otros factores. Descomponiendo el trabajo en base a tipos específicos, es posible tratar distintos tipos de manera diferente y proveer mayor previsibilidad.

Por ejemplo, la comunidad de Programación Extrema desarrolló definiciones de tipos para historias de distinto tamaño. Adquirieron nombres tales como “épica” y “grano de arena”. Una épica puede ser una historia grande que puede requerir que varias personas trabajen por varias semanas para desarrollarla, mientras que un grano de arena puede ser una historia pequeña que podría completarse por un solo desarrollador o un par de desarrolladores en unas horas. Adoptando esta nomenclatura—Épica, Historia y Grano de Arena—ahora tenemos tres tipos. Para cada tipo individual, la dispersión de variación será menor que la dispersión si todo el trabajo hubiera sido tratado como una historia.

Dentro de un departamento de desarrollo de software habitual puede haber varios tipos de trabajo. Puede haber nuevo trabajo valorado por el cliente con un nombre tal como “característica”, “historia”, o “caso de uso”. Como ya se describió, estos tipos pueden estar estratificados en elementos de tamaño, o en base a algún subtipo de dominio o perfil de riesgo. Puede haber trabajo para remover defectos, tales como “defecto de producción” o “defecto (interno)”. Puede haber trabajo de mantenimiento descrito como “refactorización”, “re-arquitectura”, o simplemente “actualización”. Los sistemas operativos de software, las bases de datos, las plataformas, los lenguajes, las APIs y las arquitecturas de servicio cambian con el tiempo y la base de código necesita ser actualizada para tratar los cambios.

Utilizando técnicas para identificar distintos tipos de ítem de trabajo, podemos cambiar la media y dispersión de variabilidad y mejorar la previsibilidad en el sistema para cualesquier tipo de trabajo.

Una estrategia adicional para mejorar la previsibilidad es asignar capacidad total de WIP por tipo específico. Por ejemplo, con mi equipo de mantenimiento en Corbis, solamente dos ítems de mantenimiento de TI eran permitidos en cualquier momento. Esto limitó la capacidad utilizada en actualizaciones de APIs y de bases de datos. Esta estrategia es particularmente útil cuando los tipos están divididos por tamaño o esfuerzo requerido, tal como una épica, una historia y un grano de arena. Asignando capacidad específica a cada tipo, la respuesta del sistema se mantiene y la previsibilidad es mayor.

Considere un equipo con un tablero de Kanban en el cual hay un límite de dos épicas, cuatro historias regulares y cuatro granos de arena. Dos épicas están en progreso. Un espacio se hace disponible en la cola para una historia regular pero no hay ninguna en el *backlog* inmediato listo para comenzar. El equipo tiene la elección de ya sea comenzar una épica o un grano de arena, o apegarse al tipo de asignación e incurrir en tiempo de ocio.

Si comienzan la épica y varios días después una historia regular llega al *backlog*, no podrán comenzar la historia regular por un buen monto de tiempo. Esto incrementará la dispersión del tiempo de espera de las historias regulares.

Iniciar un grano de arena, más pequeño, es una mejor elección pues puede ser terminado antes de que otra historia regular esté lista para comenzar. En ese caso no hay impacto pero hay un beneficio de throughput adicional. Sin embargo, si no tienen suerte y fallan en completar el ítem más pequeño antes de que la historia esté lista para comenzar, entonces la variación del tiempo de entrega de las historias regulares será afectada de forma adversa aunque no tan mal como en el escenario de la épica.

La previsibilidad y la gestión de riesgos habitualmente deben tomar la oportunidad para incrementar throughput, debido a que los propietarios del negocio y los directores senior valoran previsibilidad más que throughput. La previsibilidad genera y mantiene la confianza, un valor ágil central, mejor de lo que se logaría entregando más con menor confiabilidad.

Mezcla de clases de servicio

Si consideramos la clase de servicio descrita en el capítulo 11, podemos anticipar cómo la variabilidad puede ser afectada por la mezcla de ítems. Si una organización sufre de muchas solicitudes expreso, ellas harán todo lo demás aleatorio, incrementando el tiempo medio de espera y la dispersión de variación, lo cual reduce previsibilidad en todo el sistema.

Solicitudes de tipo Expreso son esencialmente variaciones externas y son descritas en la siguiente sección.

Si la demanda por otras clases de servicio es bastante estable, el rendimiento del tiempo de entrega de cada clase deberá ser bastante confiable. La media y la dispersión de variación deben ser medibles y deben permanecer relativamente constantes. Esto provee previsibilidad. Usted puede lograr esto si el *backlog* es lo suficientemente grande y está lleno de una mezcla fuerte de cada clase. Asigne un límite de WIP a cada clase de servicio. Esto permitirá que la media y la dispersión de cada clase se asienten y el sistema sea previsible.

Si la demanda es variable—por ejemplo, si hay tan solo unos cuantos ítems de fecha de entrega fija y tienden a ser estacionales—debe tomarse alguna acción para ya sea darle forma o controlar la demanda: Se deben instituir cambios de forma estacional en la asignación de límites de WIP a través de tipos para anticipar cambios de demanda, o alternativamente cambios en las políticas de arrastre asociadas con el conjunto de clases de servicio que llegan deben hacerse estacionalmente para tratar las fluctuaciones en la demanda.

Considere un equipo con un límite de WIP de 20, asignado como 4 ítems de fecha fija, 10 ítems de clase estándar y 6 ítems de clase intangible. Usted puede tener una política que indique que es necesario apegarse estrictamente a estos límites, o usted puede aflojar la regla y permitir que un ítem estándar o intangible llene el espacio de un ítem de fecha fija cuando no hay suficiente demanda estacional para ítems de fecha fija. Estas políticas pueden ser intercambiadas en diferentes tiempos del año para mejorar el resultado económico y asegurarse de que el sistema se mantiene bastante previsible.

Flujo irregular

El flujo irregular de trabajo puede ser causado tanto por fuentes externas como internas de variabilidad. Cada ítem simple que es arrastrado a través de un sistema kanban será diferente: diferente en naturaleza hasta cierto punto y diferente en tamaño, complejidad, perfil de riesgo y esfuerzo requerido. La aleatoriedad natural de esto causará altas y bajas en el flujo. Un sistema kanban trata naturalmente con esto siempre y cuando el límite del WIP sea reforzado. Sin embargo, mayor variabilidad de otras fuentes, tales como el tamaño del ítem de trabajo, patrones de demanda, mezcla de tipo, mezcla de clase de servicio y fuentes externas, requieren de un buffering adecuado para absorber las altas y bajas en el flujo a través del sistema. Se pueden requerir buffers adicionales—y los límites del WIP necesitarán ser más largos—donde hay mayor variabilidad en el sistema. Límites del WIP mayores resultarán en mayores tiempos de espera, pero el flujo más suave deberá reducir la variabilidad. Por lo tanto, incrementar el límite de WIP para suavizar el flujo incrementará el tiempo de entrega media y reducirá el rango de la variabilidad del tiempo de entrega. Esto es generalmente un resultado más deseable para los directores, los dueños y usualmente los clientes quienes valoran la previsibilidad más que la oportunidad aleatoria de un tiempo de entrega más corto o un mayor throughput.

Re-trabajo

El re-trabajo, ya sea debido a defectos internos que están siendo resueltos antes de la entrega o debido a defectos en producción que están desplazando nuevo trabajo valorado por el cliente, afecta la variabilidad. Si la tasa de defectos es conocida, medida regularmente y bastante constante, el sistema puede ser diseñado para tratarlos adecuadamente. Tal sistema será económicamente ineficiente, pero será confiable. Lo que ocasiona la falta de previsibilidad es cuando la tasa de defectos no es anticipada correctamente. El re-trabajo no planificado debido a defectos alarga los tiempos de entrega, tiende a incrementar la dispersión de variabilidad y reduce grandemente el throughput. Parece ser muy difícil planificar para una tasa de defectos específica, e.g., ocho defectos por historia de usuario y aún más saber o poder predecir su tamaño y complejidad. La mejor estrategia para la reducción de variabilidad debido a defectos es perseguir alta calidad y un número de defectos muy bajos de manera implacable.

El hacer cambios en el proceso del ciclo de vida del desarrollo de software puede afectar la tasa de defectos grandemente. El uso de revisión por colega, programación en parera, pruebas unitarias, estructura para pruebas automatizadas, integración continua (o muy frecuente), lotes pequeños, arquitecturas diseñadas limpiamente, código débilmente acoplado y diseño de código altamente cohesivo reducirán defectos grandemente. Los cambios que pueden afectar la tasa de defectos directamente y mejorar la previsibilidad del sistema están directamente bajo el control de la gerencia local y del equipo.

Fuentes Externas de Variabilidad

Las fuentes externas de variabilidad vienen de lugares que no están directamente controlados por el proceso de desarrollo de software o el método de gestión de proyectos. Algunos de ellos serán de otras partes del sistema o de la cadena de valor, tal como proveedores o clientes. Otras fuentes externas incluyen elementos del mundo físico que no pueden ser anticipados, previstos, o controlados fácilmente—por ejemplo, un equipo que falla o condiciones climatológicas adversas.

Ambigüedad de Requerimientos

Requerimientos escritos pobemente, planes de negocio mal definidos y falta de planificación estratégica, visión, o cualquier otra información para establecer contexto puede significar que un miembro del equipo no puede tomar una decisión y por lo tanto no puede completar una pieza de trabajo. Un ítem de trabajo se bloquea debido a su inhabilidad para tomar una decisión; se requiere nueva información para clarificar la situación de tal forma

que el miembro del equipo puede tomar una decisión de alta calidad, permitiendo que el trabajo-en-progreso fluya hacia su finalización.

Con el fin de reducir el impacto de esos bloqueos, el equipo y la gerencia directa necesitan implementar una administración de asuntos eficientes y un proceso de resolución, como se describe en el capítulo 20.

Conforme un equipo y una organización madura, será posible discutir el análisis y la eliminación de la causa raíz. Los asuntos bloqueados debido a requerimientos ambiguos pueden ser atendidos si influenciamos el proceso de análisis utilizado para desarrollar los requerimientos y mejoramos la capacidad y nivel de conocimiento de quienes las definen. Medidas tales como éstas habitualmente requieren la colaboración de otros departamentos y directores y el deseo de mejorar de parte del negocio.

En Corbis, en 2007, esto se logró a través de un proceso gradual. Primero, el sistema de kanban fue implementado, incluyendo el tablero visual, un sistema de Seguimiento electrónico y la transparencia que viene con eso. El negocio se involucró e interesó más y más en la actividad de desarrollo de software y en la monitorización de la eficiencia del proceso. Se generó un reporte que mostraba el número de asuntos abiertos, el número de ítems de trabajo bloqueados y el tiempo promedio para resolverlos. (ver la Figura 12.6, el reporte de ítems de asuntos y trabajo bloqueado).

Cuando un requerimiento pasó por todo el proceso hasta las pruebas de aceptación antes de ser rechazado como algo que el negocio no necesitaba, el equipo reaccionó creando un cesto de basura en el tablero y colocando el ticket en él, como se muestra en la figura 19.1. La gerencia pidió entonces un conjunto pequeño de reportes electrónicos que mostraran el trabajo que había entrado al sistema pero que había fallado al pasar por todo el proceso (ver Figura 19.2).

La combinación de transparencia, reportaje y generación de conciencia sobre el impacto y costo de los requerimientos pobres resultó en un cambio voluntario de comportamiento de parte del negocio. El reporte de basura que mostró el efecto de los requerimientos pobres inicialmente mostró entre 5 y 10 ítems por mes. Para el quinto mes estaba vacío. El negocio llegó a apreciar que siendo más cuidadosos podían evitar desperdiciar capacidad. Ellos colaboraron voluntariamente para mejorar el resultado del sistema. El efecto neto fue la eliminación de la causa raíz de las variaciones de causa assignable de los requerimientos escritos pobemente o de la información de contexto mal definida.

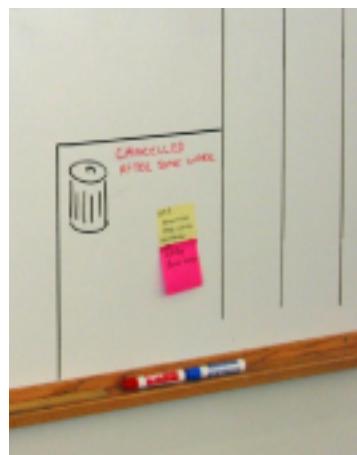


Figura 19.1 Tablero con cesto de basura

Rejected and Canceled Work Items										
Report generated 4/27/2007 4:14:05 PM by CONTINUUM DavidA; Last Warehouse update: 4/27/2007 3:31 PM										
List Bugs, CRs and PDUs either Rejected or Canceled										
ID	Work Item Type	Title	Dept	GTM Related	Business Priority	Submitted Date	Approved Date	Closed Date	Reason	
2458	Bug	A lacinia leo justo vitae massa		1-Expedite	4/5/2007			4/5/2007	Overtaken by Events	
2470	CR	Quisque vitae lacus sodales urna	Creative Services	Not Related to GTM		4/5/2007		4/5/2007	Overtaken by Events	
1463	CR	Donec posuere malesuada sodales	Media Services	Not Related to GTM	2-High	11/26/2006		4/12/2007	Released	
1443	CR	Pellentesque a. Duis et felis	Customer Experience	Not Related to GTM	2-High	11/26/2006		4/12/2007	Released	
2703	CR	Semper turpis facilisis dui	HR	Not Related to GTM		4/26/2007		4/26/2007	Canceled	

Figura 19.2 Reporte de Trabajo Rechazado y Cancelado mostrando los ítems abandonados el mes previo

A pesar de que el equipo de desarrollo de software había tomado acciones para proveer una mayor transparencia y conciencia, esas acciones no afectaron directamente el proceso de desarrollo de requerimientos. El proceso de gestión y resolución de asuntos solamente mitigó el impacto de los asuntos bloqueados incrementando la conciencia y reduciendo el tiempo de resolución. Los efectos de la transparencia y el reportaje eventualmente resultaron en un cambio externo en el proceso que eliminó la causa raíz del problema.

Ésta es evidencia anecdótica de que se pueden tomar acciones localmente que tendrán un efecto indirecto en variaciones de causa assignable.

Solicitudes Expreso

Las solicitudes expreso suceden debido a eventos externos, tales como una orden inesperada de un cliente, o debido a una cierta ruptura en el proceso interno de la empresa, por ejemplo, falta de comunicación que resulta en el descubrimiento tardío de algún requerimiento importante. Las solicitudes expreso son, por definición, variaciones de causa assignable, pues la razón de la solicitud siempre es sabida y por lo tanto siempre es “asignable”.

En la ingeniería industrial, expreso es conocido como algo malo. Afecta la previsibilidad de otras solicitudes. Incrementa el tiempo de entrega media y la dispersión de variabilidad y reduce el throughput. La evidencia colectada en Corbis a lo largo de 2007 demostró que este resultado en ingeniería industrial era cierto para procesos de desarrollo de software: acelerar es indeseable aun cuando se haga para generar valor.

La necesidad de acelerar puede reducirse. Incrementando la capacidad de holgura mediante mejoras en el throughput, automatización, o recursos involucrados mejorará la habilidad de responder. Tiempos de entrega más cortos, mayor transparencia y mejorar la madurez organizacional reducirá la necesidad de acelerar. Los equipos buenos que adoptan el enfoque de Kanban han demostrado exhibir muy poca demanda de solicitudes expreso. De hecho, durante 2007 en Corbis hubo solamente un total de cinco solicitudes de ese tipo.

Al igual que con requerimientos pobres, podemos esperar que la transparencia en el proceso e información de buena calidad respecto al throughput, tiempo de entrega y el rendimiento de la fecha límite influenciará el comportamiento de las áreas anteriores en el proceso. Esperémos que la demanda sea formada de tal manera que se entienda efectivamente lo antes posible que puede ser gestionada con una clase de servicio regular en lugar de una solicitud expreso.

Un método para provocar este cambio es acordar en limitar el número de solicitudes expreso que pueden ser procesadas en un momento dado. En Corbis, este límite era de uno. Negándole al negocio la habilidad de acelerar cualquier cosa que desee, se forzó a la gente de los niveles altos del proceso, tales como la gente de ventas y mercadotecnia, a explorar oportunidades temprano y evaluarlas efectivamente. Si a la gente de ventas se le paga en base a comisión y se le mide en base a los ingresos generados, la falta de poder acelerar algo les lastimarán. Si falló porque el límite de WIP para solicitudes expreso ha sido alcanzado, entonces intentarán juntar suficiente información para poner la solicitud a tiempo para que se logre mediante una clase de servicio regular. Nuevamente, éste es un ejemplo de una acción interna que puede ser tomada para afectar indirectamente una variación de causa assignable. Un cambio en el diseño del sistema que normalmente afectaría la variación de causa aleatoria tiene un efecto secundario en la variación externa de causa assignable.

Flujo Irregular

El flujo irregular de trabajo puede ser causado tanto por variaciones de causa assignable como por variaciones de causa aleatoria. Todas las variaciones de causa assignable que afectan el flujo dan por resultado trabajo bloqueado. Problemas tales como requerimientos ambiguos y disponibilidad de recursos compartidos ambientales o especialistas son razones comunes para el trabajo bloqueado de causa assignable.

Los ítems de trabajo bloqueado requieren de una fuerte disciplina y capacidad de gestión y resolución de asuntos, como se describe en el capítulo 20. Hay dos enfoques para tratar asuntos con ítems de trabajo bloqueado.

El primer enfoque facilitará el flujo, pero a costa del tiempo de entrega y posiblemente calidad. Usted puede mejorar el flujo teniendo un límite mayor del WIP global—logrado ya sea mediante buffering explícito o utilizando una política con menor restricción en WIP, por ejemplo, 3 cosas por persona en promedio, en lugar de 1.2 cosas por persona. El límite de WIP mayor significa que mientras algo está bloqueado, los miembros del equipo pueden estar trabajando en otros ítems. Recomiendo este enfoque para organizaciones inmaduras. Los efectos deben ser simples y sin drama. Los tiempos de entrega serán mayores, pero esto puede no ser un problema en muchos dominios. La dispersión de variación puede ser mayor por lo que los tiempos de entrega serán menos previsibles; sin embargo, serán más previsibles ahora que antes a través del uso de un sistema kanban. El mayor inconveniente

de utilizar límites de WIP mayores es que hay menos (o no) tensión para provocar discusión e implementación de mejoras. Consistentemente falta presión para mejorar: el efecto catalítico de kanban se pierde.

El segundo enfoque es aplicar la gestión y resolución de asuntos sin descanso y, conforme el equipo madura, moverse hacia el análisis y eliminación de la causa raíz con mejoras específicas diseñadas para prevenir variaciones de causa assignable en el futuro. En este enfoque usted mantiene los límites de WIP, los tamaños de los buffers y las políticas de trabajo y hace que el trabajo se detenga cuando las cosas son bloqueadas. El tiempo de ocio para las personas asignadas al trabajo bloqueado incrementa la conciencia sobre el asunto que está bloqueando. Puede ocasionar un comportamiento de enjambramiento para tratar de resolver el problema, el cual se ha visto alentar a esos miembros ociosos a pensar acerca de las causas raíz y posibles cambios en el proceso que reducen o eliminan la posibilidad de recurrencia. Se ha visto que manteniendo los límites de WIP fuertes y buscando la gestión y resolución de asuntos como una capacidad genera una cultura de mejora continua. Yo vi esto por primera vez en Corbis en 2007, pero han habido algunos otros reportes que emergieron en 2009 en firmas tales como Software Engineering Professionals en Indianápolis, IPC Media y la BBC Worldwide, ambas en Londres. Se tiene ahora suficiente evidencia que sugiere que Kanban de hecho provoca una cultura que está enfocada en mejora continua. Los elementos del proceso consistentes en los ejemplos parecen ser un deseo de reforzar políticas estrictas de WIP, de marcar el trabajo como bloqueado, de permitir que la línea se detenga, de incurrir en tiempo de ocio y de perseguir la resolución y gestión de asuntos como una disciplina organizacional. Lo que resulta de esto es un enfoque en el análisis y eliminación de la causa raíz y la introducción gradual de mejoras que tanto reducen las variaciones de causa assignable y encienden una cultura más amplia de mejora continua.

Disponibilidad del Entorno

La disponibilidad del entorno es un asunto habitual de causa assignable que puede tener un efecto significativo en el flujo, throughput y previsibilidad. Interrupciones del entorno continuas hacen que los flujos de trabajo completos se estanquen. Un sistema kanban dará visibilidad al problema y su impacto. Se ha visto que el tiempo de ocio en que se incurre haciendo cumplir un límite de WIP fomenta la colaboración para resolver la interferencia. Cuando las personas en los niveles anteriores del proceso, tales como desarrolladores y ingenieros de pruebas ayudan a la gente de mantenimiento de sistemas a recuperar el entorno, a este comportamiento se le conoce como enjambrarse. Enjambrarse implica el concepto de que el equipo se une para trabajar en un solo problema hasta que es resuelto. La naturaleza de Kanban alienta a los equipos a enfocarse en el tiempo de entrega, throughput y flujo a través de la cadena de valor. Alineando todos los grupos arriba y debajo de la cadena de valor con las mismas metas, hay una iniciativa para que emerja el comportamiento de

enjambre. Todos ganan cuando las personas con ocio se ofrecen como voluntarios para colaborar y resolver un asunto que les afecta aun cuando no es en su área de trabajo inmediata o su área de responsabilidad.

Otros Factores de Mercado

En octubre de 2008, siguiendo el colapso de Lehman Brothers y una serie de eventos traumáticos relacionados en el sector financiero, bancos y firmas de inversión en centros financieros líderes tales como Londres y Nueva York comenzaron a cancelar o a modificar significativamente sus proyectos de TI en desarrollo. La razón fue que su mundo se volvió al revés. Estaban luchando por su supervivencia. Repentinamente necesitaban entender su liquidez y la de su mercado. Dejó de ser importante entregar los productos más reciente de comodidad exótica. El mercado no le podían importar menos las inversiones. En el otoño de 2008, las empresas financieras estaban interesadas únicamente en la solvencia o insolvencia, dependiendo de cómo les fuera la suerte.

Este es un ejemplo severo—pero muy real—que muestra como las carteras y requerimientos para los proyectos en progreso pueden cambiar dramáticamente. Reaccionar a estos tipos de cambios tiende a distraer a los equipos y resultar en caídas significativas de throughput, incrementos dramáticos en los tiempos de espera, caídas (continuas) en la calidad y pérdida de previsibilidad conforme el equipo se recupera de la aleatoriedad que una fluctuación en el mercado le causa al trabajo interno del proyecto.

Claramente tales eventos son variaciones de causa assignable. Necesitan tener cabida utilizando estrategias y tácticas de gestión de riesgos. Hay un cuerpo de conocimiento considerable sobre variación de causa assignable, o riesgos basados en eventos. Construir una capacidad de gestión de riesgos fuerte como parte de una meta global para mejorar la madurez organizacional mejorará la previsibilidad de una función de ingeniería de software, ya sea utilizando o no Kanban. Sin embargo, los sistemas kanban exhiben mayor previsibilidad cuando los riesgos son bien gestionados. Esto genera mayor confianza en el sistema.

Los sistemas kanban tienen otros elementos que asisten en la gestión de riesgos. El límite de WIP reduce riesgos debido a que solamente una fracción pequeña de trabajo está en progreso en un momento dado. La asignación de límites de WIP a través de los tipos y clases de ítem de trabajo ayudan a gestionar el riesgo y absorben variaciones de causa assignable. Otras estrategias están emergiendo y es probable que un libro subsecuente detallando métodos avanzados para mejorar Kanban y tácticas para mejor gestión de riesgos también emerja.

He presentado algún material sobre gestión de riesgos con sistemas kanban que emergieron del uso de Kanban en conferencias durante 2009, lo cual está disponible en línea²⁵.

Dificultad Programando Actividades de Coordinación

Otra fuente común de variación de causa assignable que ocasiona que el trabajo se bloquee y el flujo se haga irregular es el reto de coordinar equipos externos, personas interesadas y recursos. Una reacción frecuente a los retos de coordinación es programar reuniones con una cadencia regular. En algunas ocasiones esto es muy eficiente. Sin embargo, no siempre será posible.

El flujo puede ser interrumpido por un gobierno o una restricción regulatoria que requiere una auditoría o aprobación. La gente requerida para efectuar esta función puede no estar disponible inmediatamente o puede ser difícil de programar.

En primera instancia, la variación de causa assignable de esta naturaleza debe ser tratada incrementando conciencia y atrayendo atención hacia ella con visibilidad y transparencia. Marcando los ítems como bloqueados e incrementando visibilidad en la fuente del bloqueo, la gerencia, el equipo y la gente interesada dentro de la cadena de valor serían más conscientes del impacto de tales retos de coordinación.

Esta conciencia debe llevarnos a algunos retos de comportamiento para mejorar la situación.

Una táctica podría ser examinar al gobierno y las reglas regulatorias y decidir si todo tiene que ser tratado, aprobado, auditado, o examinado. Asumiendo que algún perfil de riesgos permite que el trabajo sea bifurcado en dos categorías que necesite que tal reunión suceda o no, ya sea el tipo de ítem de trabajo o la clase de servicio pueden ser usados para separar el trabajo. Usted puede entonces usar asignación de los límites de WIP a ambos tipos o clases para asegurar un flujo suave.

Para llevar

- ❖ El estudio de variación en procesos industriales comenzó en los años veinte con Walter Shewhart y evolucionó a través del trabajo de W. Edwards Deming, Joseph Duran y David Chambers desde mediados a finales del siglo veinte.
- ❖ Un estudio de variación y su método de análisis estadístico está en el corazón tanto de el Sistema de Producción Toyota (y por ende, Lean) así como los métodos de Seis Sigma para la mejora de procesos.
- ❖ El trabajo de W. Edwards Deming y Joseph Duran fueron una inspiración mayor para el trabajo del Software Engineering Institute en la Universidad de Carnegie Mellon y el Modelo de Madurez de Capacidad (ahora Integración del Modelo de Madurez de Capacidad, o CMMI).
- ❖ Shewhart dividió las fuentes de variación en dos categorías: aquellas internas al proceso o sistema y aquellas externas al proceso o sistema.
- ❖ A las variaciones internas se las refiere como variaciones de causa aleatoria.
- ❖ A las variaciones externas se las refiere como variaciones de causa assignable.
- ❖ Puede haber muchas fuentes de variaciones de causa aleatoria en la cadena de valor del ciclo de vida de un desarrollo de software. Ejemplos habituales incluyen el tamaño del ítem de trabajo, la clase de servicio, flujo irregular y el retrabajo.
- ❖ Existe una lista posiblemente sin fin de fuentes de variaciones de causa assignable. Ejemplos habituales incluyen la ambigüedad de requerimientos, solicitudes expreso, disponibilidad de entorno, flujo irregular, factores de mercado, factores de personal y retos en la coordinación de reuniones u otras actividades extra.
- ❖ La variación de causa aleatoria puede ser controlada mediante el uso de políticas que definen el ciclo de vida del desarrollo de software y los procesos en uso de gestión de proyectos en uso.
- ❖ Las variaciones de causa assignable pueden ser administradas mediante el uso de capacidades de gestión y resolución de asuntos así como capacidades de gestión de riesgos y pueden ser reducidas o eliminadas a través de capacidades de análisis y eliminación de causa raíz.
- ❖ Los sistemas kanban producen mejores resultados económicos cuando son acoplados a una capacidad sólida de gestión de riesgos basadas en eventos.

- ❖ Kanban también ofrece formas adicionales para gestionar riesgos tales como asignación de un límite de WIP a través de las clases de servicio y los tipos de ítem de trabajo y el uso de perfil de riesgos para separar el trabajo en tipos o clases y procesarlos adecuadamente.
- ❖ Más trabajo en estrategias avanzadas de gestión de riesgos y tácticas con Kanban continúa y será el tópico de un futuro libro.

❖ CAPÍTULO 20 ❖

Políticas de gestión de asuntos y escalamiento

Cuando el trabajo en nuestro sistema kanban se bloquea por alguna razón, se ha vuelto una convención indicarlo en la pared de tarjetas adhiriendo una nota adhesiva rosa a la tarjeta, que indique la razón del bloqueo. En sistemas electrónicos, puede haber otras maneras de indicar que un ítem de trabajo está bloqueado, tal como mostrándolo con un marco rosa. Preferentemente, las características del sistema electrónico deben permitir que la razón del bloqueo sea rastreada por separado, o los asuntos de bloqueo deben ser rastreados como ítems de trabajo de primera-clase ligados al ítem valuado por el cliente que es dependiente de la solución del asunto.

Al escribir este libro, he notado que alguna gente que intenta Kanban por primera vez se refiere a estos ítems bloqueados como cuellos de botella. Esto es incorrecto. Un ítem bloqueado puede estar obstruyendo el tubo y restringiendo el flujo, pero no es un cuello de botella tal y como se describe en el capítulo 17. No es un recurso de capacidad restringida ni un recurso con disponibilidad no instantánea. De la misma manera, un corcho en un cuello de botella no es un cuello de botella. Si usted desea restaurar el flujo en la botella simplemente remueve el corcho.

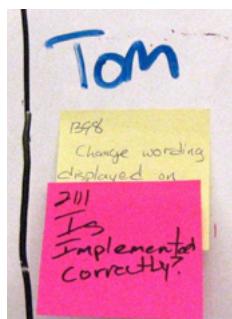


Figura 20.1 Ítem de asunto de bloqueo rosa (o impedimento) adherido al ítem de trabajo de Solicitud de Cambio directamente afectado

Es peligroso pensar sobre ítems de trabajo bloqueado como cuellos de botella porque nos lleva a un modo de pensar erróneo para resolver el problema. Los ítems de trabajo bloqueados deben ser tratados como un variaciones de causa especial en lugar de como cuellos de botella. Lo que es similar es el resultado deseado. En ambos casos—un cuello de botella o un ítem de trabajo bloqueado—deseamos resolver el asunto con el fin de mejorar el flujo.

Ítems de trabajo bloqueado requieren una organización para desarrollar la capacidad de resolución y gestión de asuntos para restaurar flujo lo más rápidamente posible, así como una capacidad de análisis de causa raíz y resolución para prevenir recurrencia. La última capacidad se discutió en el capítulo 19 como eliminación de variaciones de caso especial. El primero está discutido aquí.

Gestionando Asuntos

No es lo suficientemente bueno simplemente marcar y rastrear trabajo como bloqueado. Muchas herramientas iniciales de desarrollo de software ágil permitían tan solo esta capacidad. Mientras que es útil saber que un ítem, historia, característica o caso de uso está bloqueado mi observación de equipos alrededor del mundo ha visto que saber que algo está bloqueado no nos lleva a desarrollar una capacidad fuerte para desbloquearlo.

Es esencial rastrear la razón del bloqueo y tratarlo como un ítem de trabajo de primera clase, aunque sea un ítem de trabajo de falla de carga. Un tipo de ítem de trabajo especial, Asunto, se aparta para este propósito. Asuntos son rastreados utilizando tickets rosa (ver Figura 20.1). Se les debe asignar un número de Seguimiento y un miembro del equipo—usualmente el director de proyecto—debe ser asignado para asegurar su resolución.

Cuando un miembro del equipo que está trabajando en un ítem valuado por el cliente no puede proceder, él o ella debe marcar el ítem como bloqueado, adherirle un ticket rosa describiendo la razón del bloqueo y crear un ítem de trabajo del Asunto en la herramienta de Seguimiento electrónico. El asunto debe estar ligado a su ítem de trabajo original (ver Figura 20.1). Algunos ejemplos son: ambigüedad de requerimientos y una persona con los conocimientos no está disponible para resolver la ambigüedad inmediatamente; se requiere establecer un entorno y un ingeniero para efectuar esta tarea y no está disponible; o un especialista es requerido para trabajar en el ítem y esa persona no está disponible debido a vacaciones, enfermedad, u otro tiempo fuera-de-oficina.

Como se discutió en el capítulo 7, el mantenimiento del flujo debe ser el foco de discusión principal de la reunión de pie diaria. Por ello, la reunión se debe enfocar en discutir bloqueos y el progreso hacia la resolución de asuntos. La reunión se debe de enfocar grandemente en los tickets rosa. Se deben hacer preguntas sobre quién está trabajando en la resolución del asunto y el estado del progreso de la resolución. ¿Se tiene que escalar el asunto? ¿De ser así, a quién?

Los miembros del equipo que tienen holgura deberían ser animados para que voluntariamente rastrear los asuntos y generalmente enjambrarse en los problemas y asistir de cualquier manera posible para resolverlos y restaurar el flujo del sistema. Un equipo con capacidad fuerte de auto-organización tenderá a hacer esto naturalmente. Los miembros del equipo ayudarán voluntariamente a resolver los asuntos. Donde tal capacidad de auto-organización aún está por emerger, el director de proyecto puede necesitar asignar miembros del equipo para trabajar asuntos hasta su resolución.

Al igual que otros ítems de trabajo, los asuntos deben ser rastreados. El Seguimiento debe incluir fecha de inicio, fecha de terminación y un enlace a todos los ítems de trabajo valorados por el cliente que resulten afectados. Note que un solo asunto puede estar bloqueando más de un ítem valorado por el cliente. Esta es otra buena razón para rastrear asuntos como ítems de trabajo independientes y para tener Asunto como un tipo de ítem de trabajo distintivo. Al escoger una herramienta electrónica para rastrear su sistema kanban, asegúrese de elegir uno que soporte el seguimiento de asuntos como un tipo de primera-clase o una herramienta que se puede personalizar lo suficiente para permitirle crear un tipo de trabajo llamado Asunto y designar que será desplegado utilizando tarjetas rosas (o rojas).

Escalando Asuntos

Cuando el equipo no puede resolver un asunto por sí mismo, o alguien externo se requiere para resolver un asunto y no está disponible o no responde, el asunto debe ser escalado a un director más senior o a otro departamento.

Es importante para la organización desarrollar una capacidad muy fuerte de escalamiento de asuntos. Sin él, mantener y restaurar el flujo después del bloqueo puede ser problemático.

El fundamento de una buena capacidad de escalamiento es una política o proceso de escalamiento documentado. El capítulo 15 discute el poder de desarrollar políticas organizacionales de manera colaborativa. Las políticas de escalamiento deben ser desarrolladas de manera colaborativa y se debe llegar a un consenso entre departamentos involucrados en la cadena de valor. Las políticas de escalamiento deben ser conocidas y entendidas ampliamente y debe haber un documento (o sitio web) que las describe debe estar fácilmente disponible para todos los miembros del equipo. No debe haber ambigüedad sobre cómo y dónde escalar un problema. Dedicando tiempo para definir rutas de escalamiento y escribir las políticas a su alrededor, el equipo sabe dónde enviar asuntos para su resolución. Esto ahorra el tiempo que toma definir a quién escalar un asunto y establece expectativas para aquellos individuos más senior que se espera que sean parte del proceso. Directores senior necesitan tomar responsabilidad para resolver asuntos. Esto ayudará a mantener flujo y ultimadamente a minimizar costos de retraso (u optimizar la rentabilidad de entrega rápida).

Rastreando y Reportando Asuntos

Como se indicó anteriormente, los asuntos deben ser rastreados como ítems de trabajo de primera clase con su propio tipo de ítem de trabajo. La convención ha evolucionado a usar tarjetas rosas o rojas, o notas adhesivas para visualizar asuntos (ver Figura 20.2). Los requisitos mínimos para un sistema de seguimiento de asuntos (ver la figura 20.3) son que disponga de la fecha de entrada, la fecha de salida, un miembro del equipo asignado, la descripción de un asunto y enlaces a los elementos valorados por el cliente y que están bloqueados. Otra información que también se puede seguir es algo de historia de los esfuerzos hechos para la resolución, una historia de los individuos asignados, una indicación de la ruta de escalamiento, un tiempo estimado de resolución, una evaluación del impacto y de las soluciones sugeridas de la causa raíz para su prevención futura.

Aunque tickets rosa en la pared de tarjetas proveen una visualización fuerte de cuantos ítems están actualmente bloqueados, es también útil rastrear y reportar asuntos de otras maneras. Un diagrama de flujo acumulativo de asuntos e ítems de trabajo bloqueado provee un indicador visual fuerte de la capacidad organizacional sobre gestión y resolución de asuntos. La tendencia de los ítems de trabajo bloqueado sobre tiempo indica si una capacidad de análisis y resolución de causa raíz—oportunidades de mejora para eliminar variaciones de causa assignable—se está desarrollando. Un reporte tabular de asuntos actuales, individuos asignados, status, fecha de resolución anticipada, ítems de trabajo afectados, e impacto potencial también pueden ser útiles para la gestión diaria de proyectos grandes.

Estos reportes deben ser presentados en cada revisión de operaciones y debe asignarse tiempo para discutir la emergencia y madurez de la capacidad organizacional de gestión y resolución de asuntos y del análisis y resolución de la causa raíz. La organización debería estar consciente del impacto de la falla de carga de los asuntos bloqueados. Esto permitirá decisiones objetivas sobre oportunidades de mejora y los posibles beneficios de la inversión en resolver la causa raíz para prevenir variaciones de causa especial.

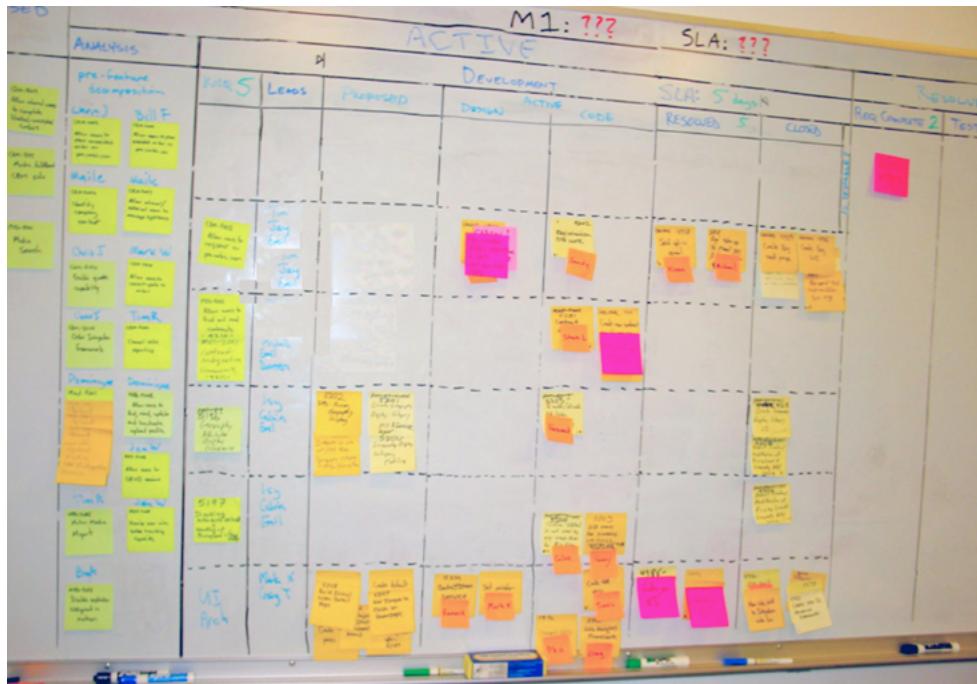


Figura 20.2 Tablero mostrando varios asuntos bloqueados afectando múltiples características

¿Cuántos asuntos e ítems de trabajo bloqueados tenemos?

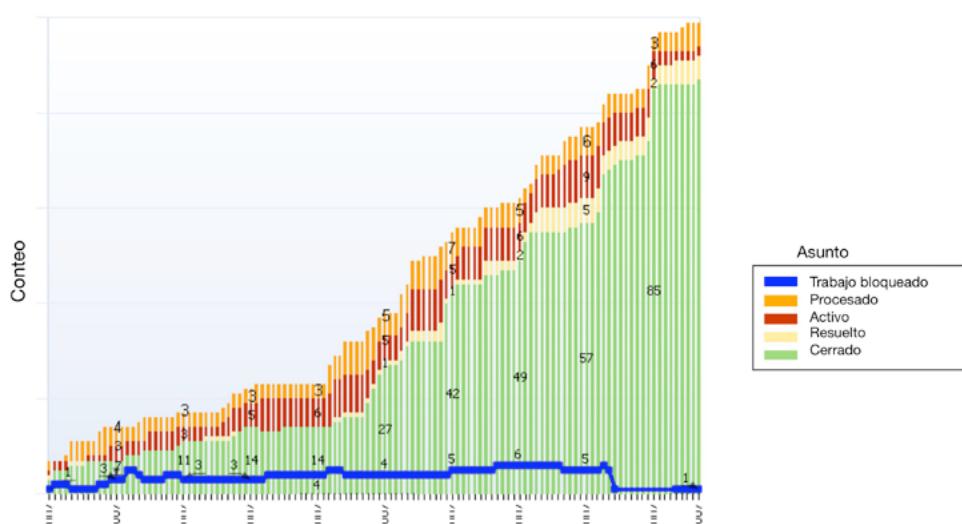


Figura 20.3 Diagrama de Flujo Acumulativo (CFD) de asuntos con gráfica superpuesta de ítems bloqueados

Para llevar

- ❖ Los sistemas kanban deben tener un Asunto tipo ítem de trabajo de primera clase y usarlo para rastrear problemas que causan que trabajo valorado por el cliente se bloquee.
- ❖ Se ha convertido en convención el utilizar notas adhesivas rosa o roja en una pared de tarjetas para visualizar asuntos de bloqueo.
- ❖ Los tickets rosa son adheridos a ítems que están bloqueados.
- ❖ Una capacidad fuerte para la gestión y resolución de asuntos es esencial para mantener flujo.
- ❖ Los ítems de trabajo bloqueados y asuntos no son cuellos de botella. Deben ser administrados como variaciones de causa especial en lugar de como recursos de capacidad restringida o recursos de disponibilidad no instantánea.
- ❖ La gestión de asuntos debe ser un enfoque fuerte de las reuniones de pie diarias.
- ❖ Una capacidad fuerte para escalamiento de asuntos es esencial como parte de una capacidad fuerte para la gestión de asuntos.
- ❖ Las políticas de escalamiento deben estar claramente definidas y documentadas y todos los miembros del equipo deben ser conscientes de ellas.
- ❖ Las políticas de escalamiento funcionan mejor cuando son acordadas en colaboración por parte de todos los departamentos involucrados en la cadena de valor.
- ❖ Los asuntos deben ser rastreados electrónicamente.
- ❖ Algún reportaje basado en datos electrónicos facilitará la gestión y resolución diaria de asuntos en proyectos grandes.
- ❖ Utilice un diagrama de flujo acumulado de Asuntos e Ítems de Trabajo Bloqueados para visualizar el desarrollo de capacidad para la gestión y resolución de asuntos, análisis y resolución de la causa raíz.

❖ CAPÍTULO 21 ❖

Kanban en Latinoamérica

por Masa K Maeda

Kanban en México

Comencé a viajar frecuentemente de Silicon Valley, California, a la Ciudad de México en enero de 2009, tres meses después de que decidí ampliar el plan de negocios de Shojiki Solutions, mi empresa, para incluir a Latinoamérica. Comencé dando un par de pláticas sobre Ágil y Lean como parte de mi estrategia de darme a conocer pues en ese entonces contaba con tan solo dos contactos en México. Me llevé una sorpresa inesperada. Aproximadamente el 90% de las audiencias tenían cero conocimiento sobre Ágil, y mucho menos sobre Lean. Reaccioné inmediatamente y en lugar de buscar negocio me dedique a dar pláticas en todo lugar posible para crear conciencia sobre Lean-Agile, y para Abril había dado ya 17 presentaciones para diversas industrias, grupos, entidades educativas y empresas específicas (a la fecha he dado no menos de 40 presentaciones). Para diciembre de ese mismo año fundé el capítulo México del APLN (Agile Project Leadership Network) con la visión de promover tanto Ágil como Lean.

Conforme pasó el tiempo conocí a un grupo pequeño de personas muy interesadas en Ágil. Algunas de ellas ya sea practicándolo—personas tales como Sergio Durán Rubio, Carlos Mondragón, Adriana Midence, Elizabeth Rivera, y Matt Pérez—o bien promoviéndolo—entre ellos Alejandro Escamilla, Francisco Lozada, Cuitláhuac Osorio, y la revista mexicana Software Gurú. Y un grupo aún más pequeño enfocado en Kanban y en Lean: Luis Nava, Elizabeth Rivera, Jesús Reynaga, la

empresa Certum, y de la CUAED/CATED Francisco Cervantes Pérez, Luis Nava y Jorge Polo; y otros entusiastas. Si hoy en día le preguntamos a ejecutivos y otro personal de TI sobre Ágil y sobre Lean será más fácil encontrar personas con por lo menos una idea acertada al respecto. Un aspecto que vale la pena mencionar es el hecho de que ha sido muy difícil el introducir estas prácticas en México debido a la tendencia fuerte a seguir regulaciones provenientes de entidades de alto nombre tales como el SEI, ISO, Prosoft etc., quienes promueven CMMI, TSP/PSP, ISO y MoProSoft entre otros. Salvo algunas excepciones, existen ejecutivos de la industria privada y de gobierno que consideran Lean, Agile, y Kanban como una curiosidad o una moda temporal debido a que no tiene el nivel de fama de las organizaciones ya mencionadas. Deciden ignorar el valor significativo que Kanban y Lean-Agile puede agregar a sus organizaciones ya sea en adición a lo que ya practican (es posible hacer Lean-Agile en conjunto con otros modelos y estándares 26, 27) o aplicándolos de manera exclusiva. La adopción de manera exclusiva es una mucho mejor alternativa debido a que separa a la empresa de la necesidad de cumplir con aspectos de los procesos que generan fricción.

Un aspecto que vale la pena mencionar es el hecho de que ha sido muy difícil el introducir estas prácticas en México debido a la tendencia fuerte a seguir regulaciones provenientes de entidades de alto nombre tales como el SEI, ISO, Prosoft etc., quienes promueven CMMI, TSP/PSP, ISO y MoProSoft entre otros. Salvo algunas excepciones, existen ejecutivos de la industria privada y de gobierno que consideran Lean, Agile, y Kanban como una curiosidad o una moda temporal debido a que no tiene el nivel de fama de las organizaciones ya mencionadas. Deciden ignorar el hecho de que agile se inició hace casi 20 años y lean hace 6 años, y el valor significativo que Kanban y Lean-Agile puede agregar a sus organizaciones ya sea en adición a lo que ya practican (es posible hacer Lean-Agile en conjunto con otros modelos y estándares 26, 27) o aplicándolos de manera exclusiva. La adopción de manera exclusiva es una mucho mejor alternativa debido a que separa a la empresa de la necesidad de cumplir con aspectos de los procesos que generan fricción.

Algunas empresas reaccionaron favorablemente y decidieron adoptar Kanban a partir de 2010, por lo que he trabajado con ellas tanto en Kanban como en gestión de proyectos Lean-Agile. Ha sido fabuloso ver la reacción tan positiva de las personas y organizaciones ante Kanban y el pensamiento Lean. Adicionalmente la resistencia al cambio fue prácticamente nula. Personas a todos los niveles de la organización han reaccionado de manera positiva y proactiva, y ello incrementa el nivel de éxito.

Como en todo lugar, algunas empresas tienen la tendencia a permitir regresar a los previos hábitos. Otro síntoma que he detectado es la tendencia a ignorar la generación y uso de métricas, resultando en una pobre mejora continua. Estos comportamientos se deben primordialmente a la falta de buen liderazgo.

Durante 2011 he visto una lenta y gradual tendencia al cambio y algunos líderes están surgiendo con la decisión de implementar Kanban en sus organizaciones.

Un Caso a Modo de Ejemplo

Una organización a la cual le di mentoría y entrenamiento en Kanban desarrolla tanto la tecnología como el contenido para educación a distancia a niveles de Licenciatura y de Bachillerato. Esta organización, dividida en tres grupos, se encontraba originalmente bajo muchos retos operacionales que hacían muy difícil la generación de la tecnología y el contenido con el nivel de calidad deseado y dentro de los montos de tiempo necesarios. Algunos problemas en la organización eran:

- Alta división jerárquica
- Pobre comunicación entre grupos
- Pobre comunicación entre jerarquías
- Pobre comunicación con los clientes
- Pobre calidad
- Baja moral entre los empleados
- Largo tiempo de entrega

Llevé a cabo con ellos una combinación de adopción de Kaban e Innovación de Valor. El trabajo en Kanban se llevó a cabo en tres etapas debido a restricciones de itinerario. Primeramente se efectuó el entrenamiento de manera remota utilizando su tecnología, tres semanas después se llevó a cabo la mentoría donde el enfoque fue primordialmente en Innovación de Valor y secundariamente en Kanban debido a las necesidades específicas de la organización. Los tres grupos iniciaron la generación de un mapa de flujo de valor y una pared de Kanban para toda la organización. Aunque cada grupo tiene necesidades y procesos internos distintos, la pared de Kanban inicial fue generada como un ejercicio para:

- Asegurar un conocimiento base de Kanban para toda la organización.
- Tener un lenguaje común de comunicación.
- Asegurar compatibilidad entre los tableros de Kanban de cada grupo debido a la dependencia que puede existir entre ellos dependiendo de los proyectos.

El siguiente paso fue que cada grupo generara sus tableros. Los resultados fueron fabulosos. El grupo de tecnología generó un conjunto de tableros, de los cuales uno estaba orientado a proyectos, uno a servicios de mantenimiento, y los demás eran tableros individuales que cada ingeniero generó para controlar su propio trabajo. El grupo de generó dos tableros que les permiten tanto llevar a cabo su función de manera individual como manejar la dependencia con el grupo de tecnología bajo compatibilidad total. Todos los

grupos innovaron en cuanto a la forma de comunicarse internamente, entre ellos, y con los clientes.

El tercer grupo decidió hacer su tablero demasiado único y el resultado fue tanto incompatibilidad como desvío de premisas de Kanban. Esto tuvo como resultado un desequilibrio.

Durante mi mentoría de seguimiento, tres semanas después de iniciada la adopción les ayudé a corregir al tercer grupo y les di una guía de mejoras a los otros dos grupos. La organización estaba colaborando de la mejor manera en su historia y la comunicación con el cliente mejorando continuamente. Los entregables sincronizados y con muchos menos defectos. El ambiente de trabajo mejoró mucho y la gente estaba más motivada para trabajar, entre otras cosas porque los líderes estaban también más involucrados y la toma de decisiones se llevaba a cabo de manera conjunta. Todo esto en dos meses y medio. La organización maduró del equivalente del nivel 2 de CMMI al nivel 3 de CMMI y estaba desplegando ya algunos comportamientos de nivel 4 y 5.

Kanban en el resto de Sudamérica

Las experiencias que tuve en México despertaron mucho mi curiosidad. ¿Cómo es posible que estando México tan conectado con los Estados Unidos se mantenga tan por detrás en cuestión de avance tecnológico, sobre todo cuando el costo es insignificante? El liderazgo lo tiene Brasil y su influencia se ha estado expandiendo a los países vecinos inmediatos, lo cual aprendí conversando con Guilherme Chapiewski, uno de los primeros agilistas en ese país. ¿Cómo es posible que esos países, estando tan alejados de los E.U. se encuentren mucho más actualizados que México? Curiosamente, también, Brasil es el único país sudamericano que no sufrió de recesión en los últimos años y de hecho tuvo un crecimiento económico envidiable aún para países desarrollados.

En octubre de 2010 tuve la oportunidad de participar en la conferencia Agiles2010 en la ciudad de Lima, Perú, y fue allí donde David Anderson y yo hicimos el lanzamiento oficial de la versión en español de este libro; y donde se dieron varias presentaciones sobre Kanban, entre ellas de parte de David y mía. De hecho la presentación de del juego de Kanban desarrollado por Russell Healy atrajo a poco más del 50% de los participantes a la conferencia. Un par de semanas después Agile Perú reportó que el tema más discutido y de mayor atención es Kanban y he tenido oportunidad de ver remotamente su crecimiento, incluyendo dentro de organizaciones educativas tanto enseñándolo como practicándolo. Posteriormente he tenido la oportunidad de colaborar con Gustavo Quiroz y Lennon Shiomokawa de Perú están activamente promoviendo su introducción en diversos sectores. En Argentina comencé a colaborar recientemente con Alejandra Alfonso. Simultáneamente Samuel Crescêncio de Brasil me informó que Agile tiene una comunidad grande, muy activa, y que Kanban está siendo adoptada rápidamente; por lo que ahora también estamos colaborando. Me ha sido claro que el nivel de entusiasmo y avance ha sido mucho más

acelerado y firme que en México. A finales de ese mismo mes inicié conversaciones con Marco Salas y Abdiel Ledesma de Panamá, y el resultado de su entusiasmo me motivó a formar un grupo empresarial con ellos en ese país. Esto fue una sorpresa inesperada pero sumamente agradable. El comportamiento en México, en el cual el empuje y entusiasmo es más bien aislado parece no ser común en otros países latinoamericanos. Ello puede ser una razón por la cual la economía en Sudamérica está creciendo, y porqué en Panamá también se tiene una economía más estable.

Por medio del Cutter Consortium tuve la oportunidad de llevar a cabo, junto con Gabe Piccoli y Laurie Williams, un estudio muy reciente sobre adopción de Kanban cuyos resultados serán publicados en septiembre de 2010. Los resultados fueron fabulosos y uno de los más sorprendentes fue encontrar que Kanban se está adoptando en todas las regiones del mundo. La tercera región con mayor porcentaje de adopción de Kanban resultó ser Latinoamérica, después de Norteamérica y Europa. Esto me sorprendió porque mi impresión siempre fue que Asia sería el tercero, sino el segundo. Por supuesto que debemos considerar la posibilidad de que Kanban en Asia esté a mejor nivel del mostrado y tan solo se trató de una participación baja en la encuesta. Esto indica la gran fuerza con la que Brasil y sus países vecinos están adoptando Kanban (por cierto que los datos indican mejores resultados que aquellos obtenidos con otras metodologías ágiles). Tan solo espero que México reaccione pronto y evite mantenerse fuera del avance actual.

Lo mejor de todo es el hecho de que Latinoamérica tiene una oportunidad única de tomar una función de liderazgo en la adopción y avance de Kanban, y de generar soluciones innovadoras que ultimadamente contribuirán a mejorar la situación económica de la región.

❖ Notas ❖

1. Anderson, David J. *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Upper Saddle River, NJ: Prentice Hall, 2003.
2. Beck, Kent. *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley, 2000.
3. Beck, Kent et al., “The Principles Behind the Agile Manifesto.” <http://www.agilemanifesto.org/principles.html>.
4. Goldratt, Eliyahu M. *What is this thing called The Theory of Constraints and How should it be implemented?* Great Barrington, MA: North River Press, 1999.
5. Anderson, David J., and Dragos Dumitriu, “From Worst to Best in 9 Months: Implementing a Drum-Buffer-Rope Solution in Microsoft’s IT Department,” Proceedings of the TOCICO World Conference, Barcelona, November 2005.
6. Belshee, Arlo. “Naked Planning, Promiscuous Pairing and Other Unmentionables.” 2008 Agile Conference, podcast. http://agiletoolkit.libsyn.com/index.php?post_id=400364.
7. Hiranabe, Kenji. “Visualizing Agile Projects Using Kanban Boards.” *InfoQ*, August 27, 2007. <http://www.infoq.com/articles/agile-kanban-boards>.
8. Hiranabe, Kenji, “Kanban Applied to Software Development: From Agile to Lean,” *InfoQ*, January 14, 2008. <http://www.infoq.com/articles/hiranabe-lean-agile-kanban>.

9. Augustine, Sanjiv. *Managing Agile Projects*. Upper Saddle River, NJ: Prentice Hall, 2005.
10. Highsmith, Jim. *Agile Software Development Ecosystems*. Boston: Addison-Wesley, 2002.
11. The Nokia Test is attributed in origin to Bas Vodde, described here by Jeff Sutherland, who has adopted and updated it. <http://jeffsutherland.com/scrum/2008/08/nokia-test-where-did-it-come-from.html>.
12. Beck et al., “The Principles Behind the Manifesto.” <http://www.agilemanifesto.org/principles.html>.
13. Jones, Capers. *Software Assessment Benchmarks and Best Practices*. Boston: Addison-Wesley, 2000.
14. Ambler, Scott. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. Hoboken, N.J.: Wiley, 2002.
15. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. *CMMI: Guidelines for Process Integration and Product Improvement*, 2d ed. Boston: Addison Wesley, 2006.
16. Sutherland, Jeff, Carsten Ruseng Jakobsen, and Kent Johnson. “Scrum and CMMI Level 5: A Magic Potion for Code Warriors.” Proceedings of the Agile Conference, Agile Alliance/IEEE, 2007.
- Jakobsen, Carsten Ruseng and Jeff Sutherland. “Mature Scrum at Systematic.” *Methods & Tools*, Fall 2009. <http://www.methodsandtools.com/archive/archive.php?id=95>.
17. Larman, Craig and Bas Vodde. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Boston: Addison-Wesley, 2008.
18. Willeke, Eric, with David J. Anderson and Eric Landes (editors) *Proceedings of the Lean & Kanban 2009 Conference*. Bloomington, IN: Wordclay, 2009.
19. Beck et al, Principles Behind the Agile Manifesto, 2001, <http://www.agilemanifesto.org/principles.html>
20. Anderson, David J. “New Approaches to Risk Management.” Agile 2009, Chicago, Illinois. <http://www.agilemanagement.net/Articles/Papers/Agile2009-NewApproachesto.html>.
26. Maeda, Masa K. “Gobierno Corporativo + AgilidadÑ Beneficiando Est ndares y Modelos Mediante Agile y Lean.” Revista Software Guru #25 Agosto-Octubre 2009
27. Glazer, Hillel, Dalton, Jeff, Anderson, David J., Konrad, Mike, and Shrum, Sandy. CMMI or Agile: Why not Embrace Both!” SEI Technical Note CMU/SEI-2008-TN-003

❖ Reconocimientos ❖

Todo libro publicado representa un esfuerzo significativo de gestión de proyecto y coordinación que involucra un equipo de personas completo, del cual el autor es una parte pequeña. Éste libro no habría sido posible sin los esfuerzos invaluables, el duro trabajo y la dedicación de Janice Linden-Reed y Vicki Rowland. Deseo agradecerles por su increíble paciencia y perseverancia para lograr tener el manuscrito listo para impresión aún en contra de un itinerario apretado (con un alto costo de retraso).

Deseo agradecerle a Donald Reinertsen por motivarme a usar Kanban y por darme un foro para hablar sobre él públicamente. También deseo agradecerle por sus amables palabras en el Prólogo y por sus esfuerzos para construir una comunidad de vida larga y próspera mediante la formación del Consorcio de Software y Sistemas Lean.

Deseo agradecer a Karl Scotland, Joe Arnold, y Aaron Sanders, junto con Eric Willeke, Chris Shinkle, Olav Maassen, Chris Matts y Rob Hathaway. Su entusiasmo temprano y adopción de Kanban llevó directamente a la formación de la comunidad próspera y la difusión viral del método alrededor del mundo. Sin su soporte no habría demanda para éste manuscrito, y Kanban sería un método obscuro utilizado en un par de compañías en el Noroeste Pacífico de los Estados Unidos, en lugar de ser el nuevo enfoque excitante utilizado por equipos en todos los continentes—desde empresas florecientes de cinco empleados en Cambodia hasta empresas de seguros de 300 años de existencia en los Países Bajos, empresas petroleras grandes en Brasil y empresas de outsourcing en Argentina; así como empresas de medios en Londres, Los Ángeles y Nueva York y muchas más alrededor del mundo. La adopción de Kanban es un fenómeno y no habría sucedido sin el encuentro fortuito de mentes que ocurrió en agosto de 2007 en Washington D.C. en la conferencia *Agile 2007*.

Éste libro no sería una herramienta tan útil y una lectura disfrutable sin los comentarios y retroalimentación constructiva de un grupo grande de revisores del manuscrito. Deseo hacer mención especial a las contribuciones de Daniel Vacanti, Greg Brougham, Christina Skaskiw, Chris Matts, Bruce Mount, Norvert Winklareth y, nuevamente, Janice Linden-Reed. Cada uno de ellos produjo una revisión estratégica bien pensada de una o mas de las versiones tempranas del manuscrito que llevó a una reestructuración significativa del contenido. El resultado es un libro mucho mejor que es más fácil de leer, de entender y que será una herramienta de largo plazo más útil para la comunidad.

Adicionalmente, hubo muchos más miembros de la comunidad que contribuyeron con retroalimentación y ediciones que fueron consideradas cuidadosamente conforme el manuscrito emergió y evolucionó durante los años 2009 y 2010. Gracias a Jim Benson, Matthias Bohlen, Joshua Kerievsky, Chris Simmons, Dennis Stevens, Arne Roock, Matthias Skarin, Bill Barnett, Olav Maassen, Steve Freeman, Derick Bailey, John Heintz, Lilian Nijboer, Si Alhir, Siddharta Govindaraj, Russell Healy, Benjamin Mitchell, David Joyce, Tim Uttormark, Allan Kelly, Eric Willeke, Alan Shalloway, Alisson Vale, Maxwell Keeler, Guilherme Amorim, Reni Elisabeth Pihl Friis, Nis Holst, Karl Scotland y Robert Hathaway.

Deseo agradecer a mi incansable director de oficina, Mikiko Fujisaki, quien mantiene las ruedas girando en David J. Anderson & Asociados, Inc., y sin quien jamás habría encontrado el tiempo para escribir éste libro.

Mi viejo amigo y colega Pujan Roka cordialmente ofreció dibujar la ilustración de la portada. Pujan es un artista cómico talentoso y es también un autor publicado, con dos libros significativos sobre gestión a la fecha. Aprenda más sobre él y sus publicaciones en <http://www.pujanroka.com/>

La comunidad a sido muy generosa con su adopción y entusiasmo por Kanban; y se ha extendido a los ofrecimientos generosos de traducir éste manuscrito a lenguajes locales. Deseo agradecer a Jan Piccard de Muller, Andrea Pinto, Eduardo Bobsin, Arne Roock, Masa K. Maeda y Hiroki Kondo, quienes ya están trabajando arduamente produciendo las traducciones al francés, portugués (brasileño), alemán, español y japonés. Estoy seguro que sus esfuerzos ayudarán a difundir la adopción de Kanban alrededor del mundo y a expandir la comunidad y el entusiasmo por el método en sus respectivas regiones.

Deseo reconocer a Nicole Kohari, Chris Hefley, David Joyce, Thomas Blomseth, Jeff Patton y Steve Reid, quienes contribuyeron imágenes para usarlas en el libro.

Finalmente, deseo agradecer a mi buen amigo Dragos Dimitriu, ahora con Avanade, y a mi equipo en Corbis: Darren Davis, Larry Cohen, Mark Grotte, Dominica Degrandis, Troy Magennis, Stuart Corcoran, junto con Rick Garber, Corey Ladas y Diana Kolomiyets. Sin todos ellos Kanban nunca habría sucedido. Sus esfuerzos en implementarlo y usarlo crearon los ejemplos e historias de las cuales todos hemos aprendido y subsecuentemente adoptado y adaptado soluciones para situaciones nuevas y de mayor reto. Sin ellos no habría ni libro, ni comunidad, y ni banda creciente de clientes gustosos quienes están disfrutando de

software de alta calidad desarrollado regular y rápidamente, donde y cuando lo necesitan, con agilidad, en respuesta a la demanda natural de su industria y base de usuarios.

Nuestra jornada con Kanban continua y esperanzadoramente ha sido persuadido con este libro a ser parte de ella.

David J Anderson

En la vereda de evangelización de Kanban,
en algún lugar en Europa, Abril de 2010

Reconocimientos de la versión en español

Traducir un libro es una labor muy ardua. Las dificultades incluyen no traicionar al autor, no traicionar el contenido y su esencia, no traicionar el idioma, no traicionar al lector y no traicionarse a si mismo.

Estoy profundamente agradecido primeramente a David J. Anderson por haber depositado su confianza en mi para traducir su libro y espero poder continuar colaborando con él a otros niveles.

La traducción habría sido poco exitosa sin la valiosísima ayuda de Janice Linden-Reed, Wes Harris y Mikiko Fujisaki, quienes dedicaron mucho de su tiempo y disposición para asegurarse que yo contara con los recursos necesarios para llevar a cabo mi labor. Vicki Rowland hizo una labor editorial y artística fabulosa.

Deseo agradecer a Elizabeth Rivera por haberme ayudado con una gran parte de la labor tanto discutiendo la traducción como revisando el manuscrito a nivel técnico, e inclusive tecleando cuando mis problemas de túnel carpiano eran dolorosos. Agradezco a Damián Yukio Romero Díaz contribuyó revisando el estilo y la neutralidad del estilo del español utilizado que esperanzadoramente no favorece ni traiciona ninguna versión del idioma. Mi agradecimiento va también a Miguel Miranda por ayudarme con aspectos técnicos y revisar la sección cuatro a pesar de haber tenido muy limitada disponibilidad. Mi agradecimiento especial a Ángel Águeda Barrero por una revisión muy detallada de la traducción. Finalmente, gracias a Kenji Maeda por su ayuda con la creación del índice.

Una muy buena parte de la traducción la efectué durante mis viajes de negocio a México, tanto durante los vuelos como en Starbucks, durante los tiempos intermedios entre mentorías a clientes.

Masa Kevin Maeda

Agosto 5, 2010. Starbucks de Insurgentes y Parroquia en la Ciudad de México

(nueva revisión en Septiembre 5 de 2011 en Campbell California, USA)

CEO y fundador, Shojiki Solutions

Acerca del autor

David J. Anderson es el líder de una firma de consultoría en gestión enfocada en mejorar el desempeño de empresas de tecnología. Él ha estado involucrado con el desarrollo de software por casi 30 años y ha dirigido equipos en proyectos de desarrollo de software ágil en Sprint, Motorola, Microsoft, y Corbis. David tiene el crédito por la primer implementación de un proceso de Kanban para el desarrollo de software en 2005. David fue un fundador del movimiento ágil a través de su involucro en la creación del Desarrollo Basado en Características. Él es también un fundador del *Agile Project Leadership Network (APLN)*, uno de los fundadores de la Declaración de Interdependencia, y un miembro fundador del Consorcio de Software y Sistemas Lean. Él modera varias comunidades en línea para el desarrollo *lean/agile*. Él es el autor del libro *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Más recientemente, David se ha enfocado en la creación de una sinergía entre el modelo de CMMI para madurez organizacional con los métodos *Agile* y *Lean* a través de proyectos con Microsoft y el SEI. Él es un co-autor de la nota técnica “CMMI or Agile: Why not Embrace Both!” del SEI. Él está basado en Sequim, Washington, E.U.

Recursos adicionales sobre Kanban

David J. Anderson and Associates

<http://djandersonassociates.com>

The Límited WIP Society

<http://www.límitedwipsociety.org>

Kanban Development Yahoo Group

<http://finance.groups.yahoo.com/group/kanbandev>

Kanban 101

<http://www.kanban101.com>

Kanban se está convirtiendo en una manera popular de visualizar y limitar el monto de trabajo-en-progreso en las áreas de desarrollo de software y de tecnologías de la información. Equipos alrededor del mundo están agregando kanban alrededor de sus procesos actuales para catalizar el cambio cultural y entregar con mejor agilidad de negocio.

Éste libro constesta las preguntas:

- ❖ ¿Qué es Kanban?
- ❖ ¿Por qué querría yo usar Kanban?
- ❖ ¿Cómo implemento Kanban?
- ❖ ¿Cómo reconozco oportunidades de mejora y que hago al respecto?

David J. Anderson fué el pionero de la técnica de Kanban con Microsoft en el 2004 y ha estado refinando su enfoque desde ese entonces. *Kanban* nos brinda su visión sobre este nuevo enfoque evolutivo para cambiar la gestión. El Método de Kanban mejorará la madurez y agilidad de su organización con resistencia mínima al cambio.

“Éste libro lo lleva a la segunda generación de métodos Ágiles. Predigo que se convertirá en un clásico instantáneo.”

—**Alan Shalloway**, CEO y Consultor Senior,
Net Objectives

“Éste libro tiene la utilidad inmediata de una guía de inicio, pero también tiene contenido suficiente para tenerme regresando por más.”

—**Chris Simmons**, Gerente de Desarrollo, Sophos

“Kanban cambió mi negocio. ¡Leyendo este libro ahora entiendo porqué!”

—**Alisson Vale**, Fundador, Phidelis

David J. Anderson es un consultor independiente de gestión con muchos años de experiencia dirigiendo y liderando grupos de software en algunas de las empresas más grandes del mundo. El es un fundador del Consorcio de Software y Sistemas Lean y co-autor de la nota técnica “CMMI or Agile: Why not embrace both!” del Instituto de Ingeniería de Software.



BLUE HOLE PRESS
Sequim, Washington
www.blueholepress.com