

Universidad Tecnológica Nacional
Cátedra de Ingeniería de Software
Docentes: Judith Meles y Laura Covaro

Filosofía

Lean



Judith Meles

Principios Lean

Principios

Eliminar Desperdicios

Amplificar Aprendizaje

Embeber la Integridad conceptual

Diferir compromisos

Dar poder al equipo

Ver el todo

Entregar lo antes posible



Principios Lean

- Eliminar el desperdicio: evitar que las cosas se pongan viejas antes de terminarlas o evitar re-trabajo.
 - Tiene que ver con el principio ágil de Software funcionando y el de simplicidad (arte de maximizar lo que no hacemos)

Pensamiento Lean: Siete Principios

- Eliminar Desperdicios: reducir el tiempo removiendo lo que no agrega valor.
 - Desperdicio es cualquier cosa que interfiere con darle al cliente lo que el valora en tiempo y lugar donde le provea más valor.
 - En manufactura: el inventario
 - En Software: es el trabajo parcialmente hecho y las características extra!
 - El 20 % del software que entregamos contiene el 80% del valor

Gastos en producción Lean



PRODUCCIÓN
EN EXCESO



STOCK



PASOS EXTRA
EN EL PROCESO



BÚSQUEDA DE
INFORMACIÓN



DEFECTOS



ESPERAS



TRANSPORTES



Los siete desperdicios Lean (en software)

- Características extra
- Trabajo a medias
- Proceso extra
- Movimiento
- Defectos
- Esperas
- Cambio de Tareas

Pensamiento Lean: Siete Principios

- Amplificar el aprendizaje: crear y mantener una cultura de mejoramiento continuo y solución de problemas.
 - Un proceso focalizado en crear conocimiento esperará que el diseño evolucione durante la codificación y no perderá tiempo definiéndolo en forma completa, prematuramente.
 - Se debe generar nuevo conocimiento y codificarlo de manera tal que sea accesible a toda la organización.
 - Muchas veces los procesos “estándares” hacen difícil introducir en ellos mejoras.

Principios Lean

- Embeber la integridad conceptual. Encastrar todas las partes del producto o servicio, que tenga coherencia y consistencia (tiene que ver con los Requerimientos No Funcionales). La integración entre las personas hace el producto más integro.

PENSAMIENTO LEAN: SIETE PRINCIPIOS

- Embeber la integridad conceptual: se necesita más disciplina no menos!
 - Integridad Percibida: el producto total tiene un balance entre función, uso, confiabilidad y economía que le gusta a la gente.
 - Integridad Conceptual: todos los componentes del sistema trabajan en forma coherente en conjunto.
 - El objetivo es construir con calidad desde el principio, no probar después.
 - Dos clases de inspecciones:
 - Inspecciones luego de que los defectos ocurren.
 - Inspecciones para prevenir defectos.
 - Si se quiere calidad no inspeccione después de los hechos!
 - Si no es posible, inspeccione luego de pasos pequeños.

Principios Lean

- Diferir Compromisos. El último momento responsable para tomar decisiones (en el cual todavía estamos a tiempo). Si nos anticipamos tenemos información parcial.
 - Se relaciona con el principio ágil: decidir lo más tarde posible pero responsablemente. No hacer trabajo que no va a ser utilizado. Enlaza con el principio anterior de aprendizaje continuo, mientras más tarde decidimos más conocimiento tenemos.

Pensamiento Lean

- Diferir compromisos: las decisiones deben tomarse en el último momento que sea posible.
 - No significa que todas las decisiones deben diferirse.
 - Se debe tratar de tomar decisiones reversibles, de forma tal que pueda ser fácilmente modificable.
 - Vencer la “parálisis del análisis” para obtener algo concreto terminado.
 - Las mejores estrategias de diseño de software están basadas en dejar abiertas opciones de forma tal que las decisiones irreversibles se tomen lo más tarde posible.

Principios Lean

- Dar poder al equipo: ejemplo, vamos a comer a un restaurante y no nos metemos en la cocina del restaurante. Nos fijamos en el precio, pedimos y esperamos. Hay mucho micro management, el dueño no decide cuánta sal poner a la comida.
- Respetar a la gente
 - Entrenar líderes
 - Fomentar buena ética laboral
 - Delegar decisiones y responsabilidades del producto en desarrollo al nivel más bajo posible
- Ágil: El propio equipo pueda estimar el trabajo.

Principios Lean

- Ver el todo: tener una visión holística, de conjunto (el producto, el valor agregado que hay detrás, el servicio que tiene los productos como complemento).

Principios Lean

- **Entregar rápido:** estabilizar ambientes de trabajo a su capacidad más eficiente y acotar los ciclos de desarrollo.
- **Entregar rápidamente** esto hace que se vayan transformando “n” veces en cada iteración. Incrementos pequeños de valor. Llegar al producto mínimo que sea valioso. Salir pronto al mercado.
 - Relacionado con el principio Ágil de entrega frecuente.



KANBAN

Cambio Evolutivo Exitoso Para su
Negocio de Tecnología



David J. Anderson

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

En desarrollo de
software el
referente principal
es...

KANBAN

An illustration of a Kanban board. The board is a green chalkboard with a white grid. Several colorful sticky notes (yellow, blue, pink, green) are attached to the board, each with a small circle and a horizontal line. Hands are shown interacting with the board: one hand is writing on a green sticky note with a red marker, another is pointing at a yellow sticky note, and others are moving or holding sticky notes. A white horizontal line is drawn across the middle of the board, passing through the word 'KANBAN'.

KANBAN

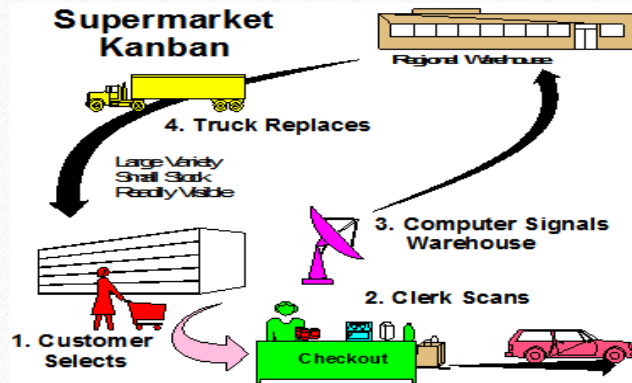
Kanban en pocas palabras

- **kan-ban** (看板) = Signal-card.

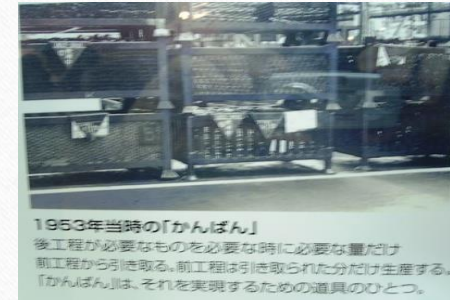


Kanban en pocas palabras-Just in Time

- A fines de 1940, Toyota comenzó a estudiar técnicas de almacenamiento y tiempo de stockeo de los supermercados



Taiichi Ohno



Kanban en pocas palabras:

Administración de Colas

- Los cajeros se focalizan en tomar órdenes.
- El Barista se focaliza en proveer café.
- Separarlos por la cola permite que se absorba la demanda variable.
- Los cajeros se mueven a ayudar al Barista cuando no hay clientes esperando para hacer su pedido.
- **Foco es en Flujo “fin a fin” FLOW = Centrado en el Cliente**

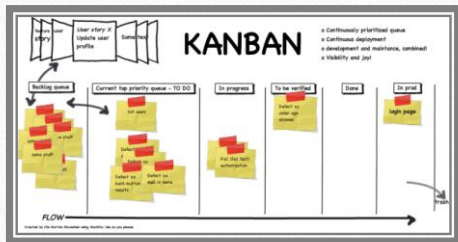


The image shows a hand-drawn Kanban board on a whiteboard. The board is organized into six columns: Product Backlog, Ready, In Progress, Ready to Test, Test, and Done. A vertical timeline on the left side of the board indicates the progression of time from 10:00 AM to 10:00 PM. The tasks are represented by sticky notes, some of which are color-coded (pink, green, yellow) and some have icons of people. The board also includes a header for 'Product Backlog' with a date '(3000)' and a 'Ready' column with a date '2023/01/01'. The 'In Progress' column has a date '2023/01/01' and a 'Ready to Test' column with a date '2023/01/01'. The 'Test' column has a date '2023/01/01' and the 'Done' column has a date '2023/01/01'.

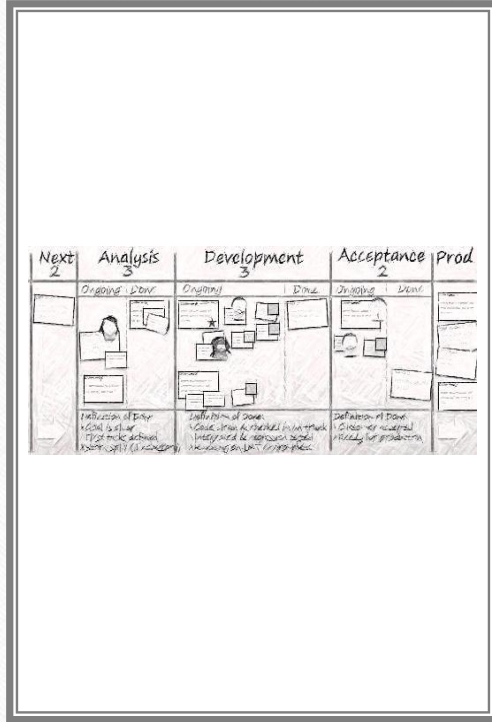
| Product Backlog | Ready | In Progress | Ready to Test | Test | Done |
|--|---|---|---|---|---|
| <p>Task 1: Implement user authentication</p> <p>Task 2: Implement user registration</p> <p>Task 3: Implement user login</p> <p>Task 4: Implement user profile management</p> <p>Task 5: Implement user settings</p> <p>Task 6: Implement user password reset</p> <p>Task 7: Implement user email verification</p> <p>Task 8: Implement user social media login</p> <p>Task 9: Implement user two-factor authentication</p> <p>Task 10: Implement user account deletion</p> | <p>Task 11: Implement user dashboard</p> <p>Task 12: Implement user profile page</p> <p>Task 13: Implement user settings page</p> <p>Task 14: Implement user password reset page</p> <p>Task 15: Implement user email verification page</p> <p>Task 16: Implement user social media login page</p> <p>Task 17: Implement user two-factor authentication page</p> <p>Task 18: Implement user account deletion page</p> | <p>Task 19: Implement user dashboard (Frontend)</p> <p>Task 20: Implement user profile page (Frontend)</p> <p>Task 21: Implement user settings page (Frontend)</p> <p>Task 22: Implement user password reset page (Frontend)</p> <p>Task 23: Implement user email verification page (Frontend)</p> <p>Task 24: Implement user social media login page (Frontend)</p> <p>Task 25: Implement user two-factor authentication page (Frontend)</p> <p>Task 26: Implement user account deletion page (Frontend)</p> | <p>Task 27: Implement user dashboard (Backend)</p> <p>Task 28: Implement user profile page (Backend)</p> <p>Task 29: Implement user settings page (Backend)</p> <p>Task 30: Implement user password reset page (Backend)</p> <p>Task 31: Implement user email verification page (Backend)</p> <p>Task 32: Implement user social media login page (Backend)</p> <p>Task 33: Implement user two-factor authentication page (Backend)</p> <p>Task 34: Implement user account deletion page (Backend)</p> | <p>Task 35: Implement user dashboard (Integration)</p> <p>Task 36: Implement user profile page (Integration)</p> <p>Task 37: Implement user settings page (Integration)</p> <p>Task 38: Implement user password reset page (Integration)</p> <p>Task 39: Implement user email verification page (Integration)</p> <p>Task 40: Implement user social media login page (Integration)</p> <p>Task 41: Implement user two-factor authentication page (Integration)</p> <p>Task 42: Implement user account deletion page (Integration)</p> | <p>Task 43: Implement user dashboard (Deployment)</p> <p>Task 44: Implement user profile page (Deployment)</p> <p>Task 45: Implement user settings page (Deployment)</p> <p>Task 46: Implement user password reset page (Deployment)</p> <p>Task 47: Implement user email verification page (Deployment)</p> <p>Task 48: Implement user social media login page (Deployment)</p> <p>Task 49: Implement user two-factor authentication page (Deployment)</p> <p>Task 50: Implement user account deletion page (Deployment)</p> |

- 29

Kanban en el Desarrollo de Software

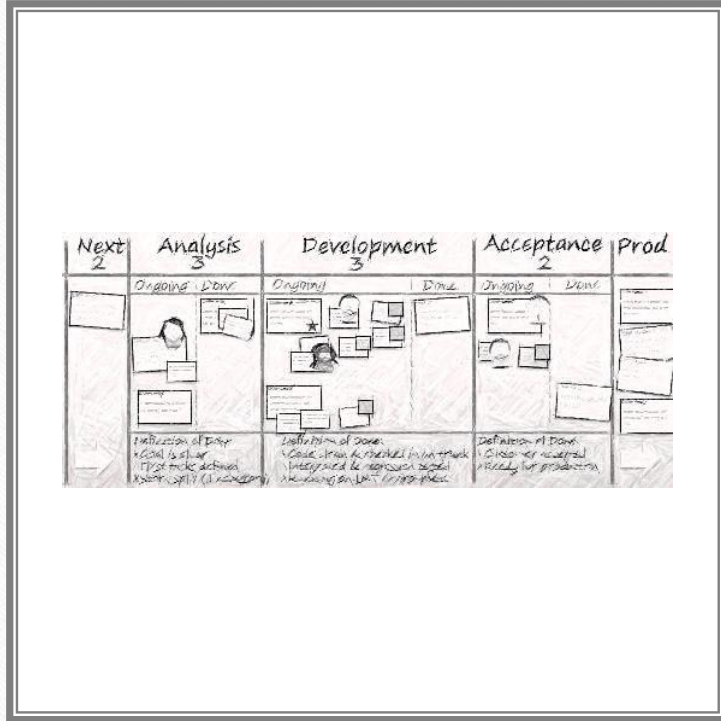


- El método fue formulado por David J. Anderson
- Es un enfoque para gestión de cambio.
- No es un proceso de desarrollo de software o una metodología de administración de proyecto.
- Kanban es un método para introducir cambios en un proceso de desarrollo de software o una metodología de administración de proyectos



Kanban en el Desarrollo de Software

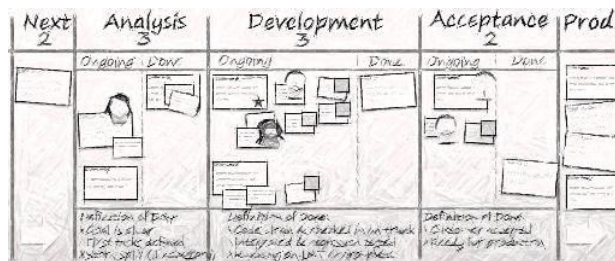
- Kanban aprovecha muchos de los conceptos probados de Lean:
 - Definiendo el Valor desde la perspectiva del Cliente.
 - Limitando el Trabajo en Progreso (WIP).
 - Identificando y Eliminando el Desperdicio.
 - Identificando y removiendo las barreras en el Flujo.
 - Cultura de Mejora Continua.



Kanban en el Desarrollo de Software

- Kanban fomenta la evolución gradual de los procesos existentes.
- Kanban no pide una revolución, sino que fomenta el cambio gradual.
- Kanban está basado en una idea muy simple: Limitar el trabajo en progreso (WIP).
- El Kanban (o tarjeta de señal) implica que una señal visual se produce para indicar que el nuevo trabajo se puede tirar ("pull") porque el trabajo actual no es igual al límite acordado.

¿Cómo aplicar Kanban?



- Empezar con lo que se tiene ahora.
- Entender el proceso actual.
- Acordar los límites de WIP para cada etapa del proceso.
- A continuación, comienza a fluir el trabajo a través del sistema tirando de él, en presencia de señales Kanban.

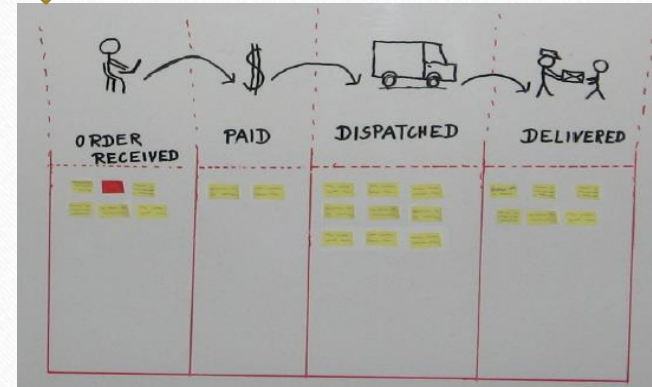
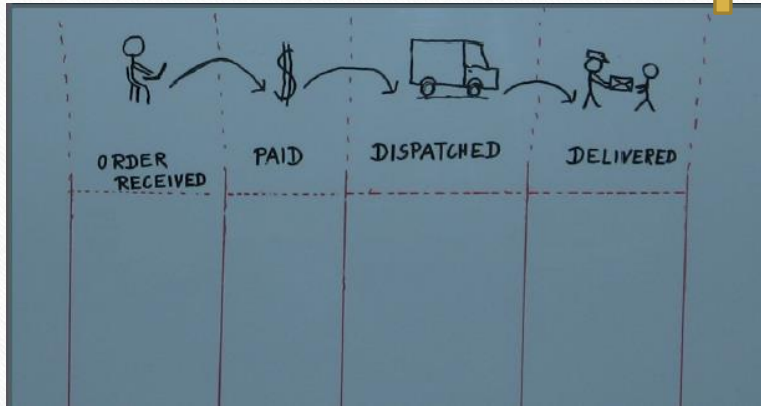
¿Cómo aplicar Kanban?

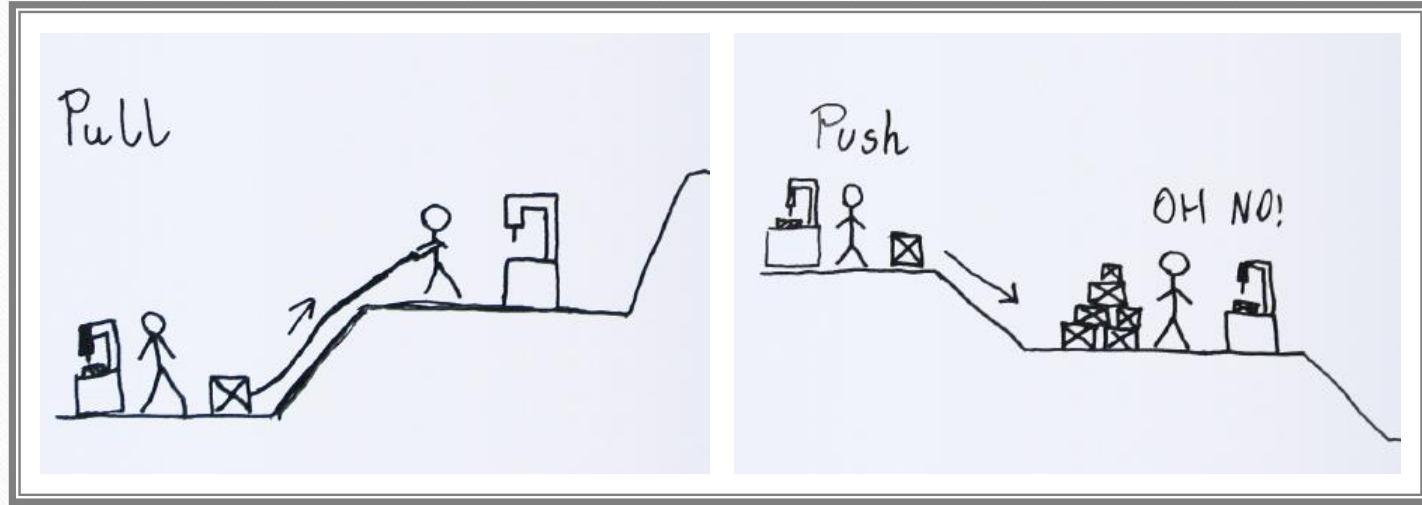
- Visualizar el flujo de trabajo:
 - Dividir el trabajo en piezas, las user stories son buenas para eso.



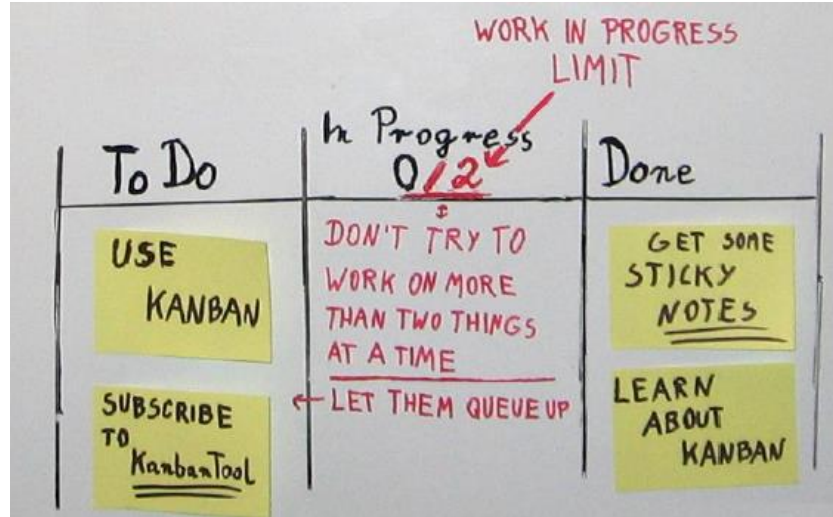
¿Cómo aplicar Kanban?

- Visualizar el flujo de trabajo:
 - Utilizar nombres en las columnas para ilustrar donde está cada ítem en el flujo de trabajo.
 - Distribuir el trabajo en las columnas: el trabajo fluirá de izquierda a derecha en las columnas.





Pull, no push !!!



¿Cómo aplicar Kanban?

- Limitar WIP – Asignar límites explícitos de cuántos ítems puede haber en progreso en cada estado del flujo de trabajo.



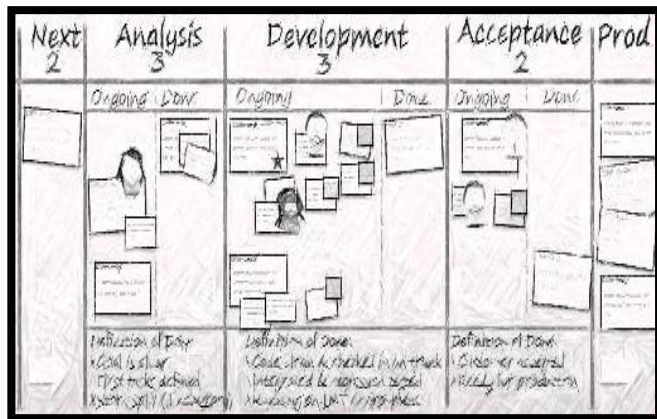
Ayudar a que el trabajo fluya....

Al 100 % de capacidad se tiene un rendimiento mínimo...

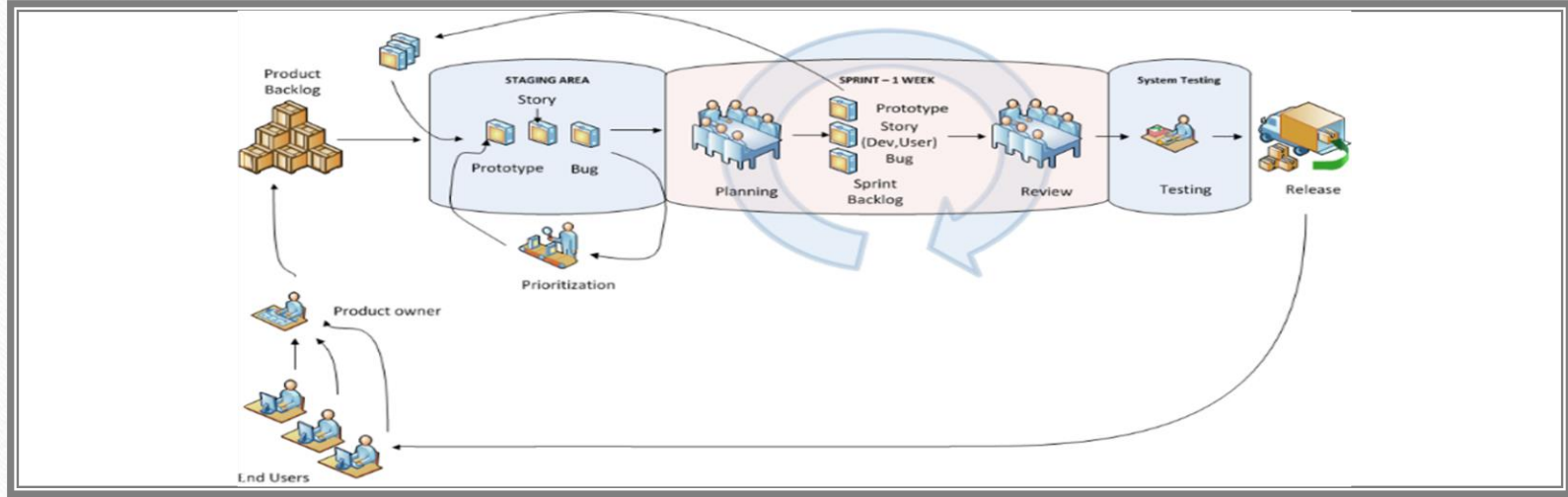
Por último, la auto asignación de tareas se ve reflejada con un avatar personalizado... :))



¿Cómo aplicar Kanban en nuestro proyecto?

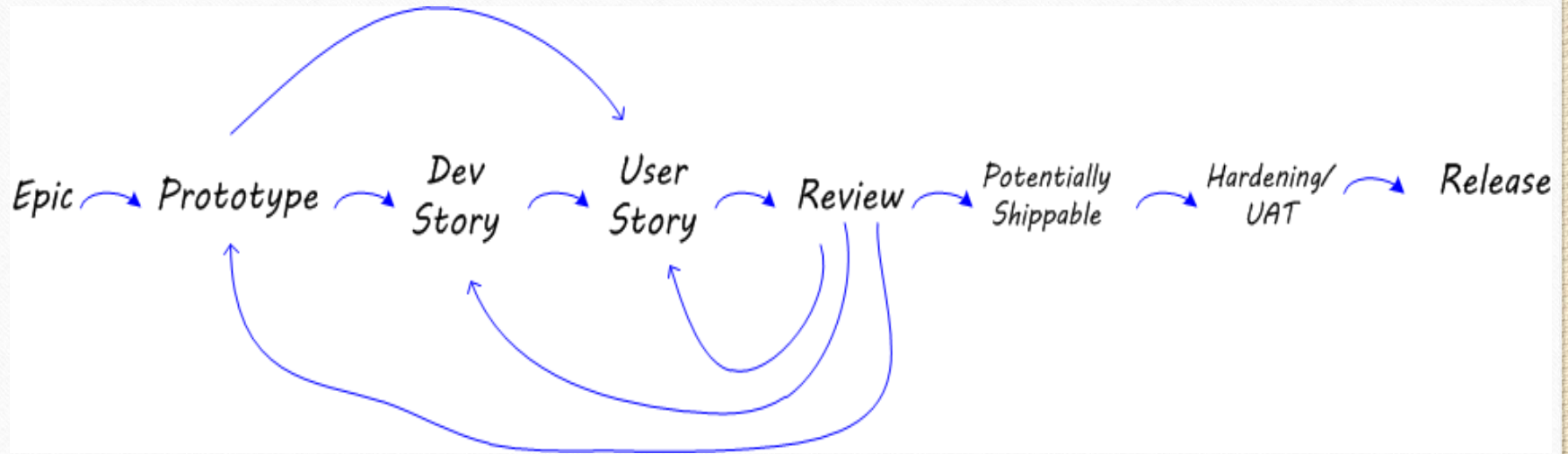


- **Proceso:** modelar nuestro proceso.
- **Trabajo :** decidir la unidad de trabajo.
- **Límites de WIP:** limitar el WIP para ayudar al flujo de trabajo.
- **Política:** definir políticas de calidad.
- **Cuellos de Botella y Flujo:** mover recursos a los cuellos de botella.
- **Clase de Servicio:** diferentes trabajos tienen diferentes políticas – definición de hecho ("done"), para cada estado.
- **Cadencia:** Releases, planificaciones, revisiones



Modelar el proceso

Cádena de Valor



Definir el proceso...

| Cola de Producto | Análisis | | Desarrollo | | Listo para Build | En Testing | | En Producción |
|------------------|-------------|-------|-------------|-------|------------------|-------------|-----------------------|---------------|
| | En progreso | Hecho | En progreso | Hecho | | En Progreso | Listo para Despliegue | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

Definir tipos de trabajo...

Asignando capacidad en función de la demanda

Requerimientos

- Caso de uso
- Historias de Usuario
- Porciones de Casos de Uso
- Características

Defectos

- Defectos en Producción
- Defectos

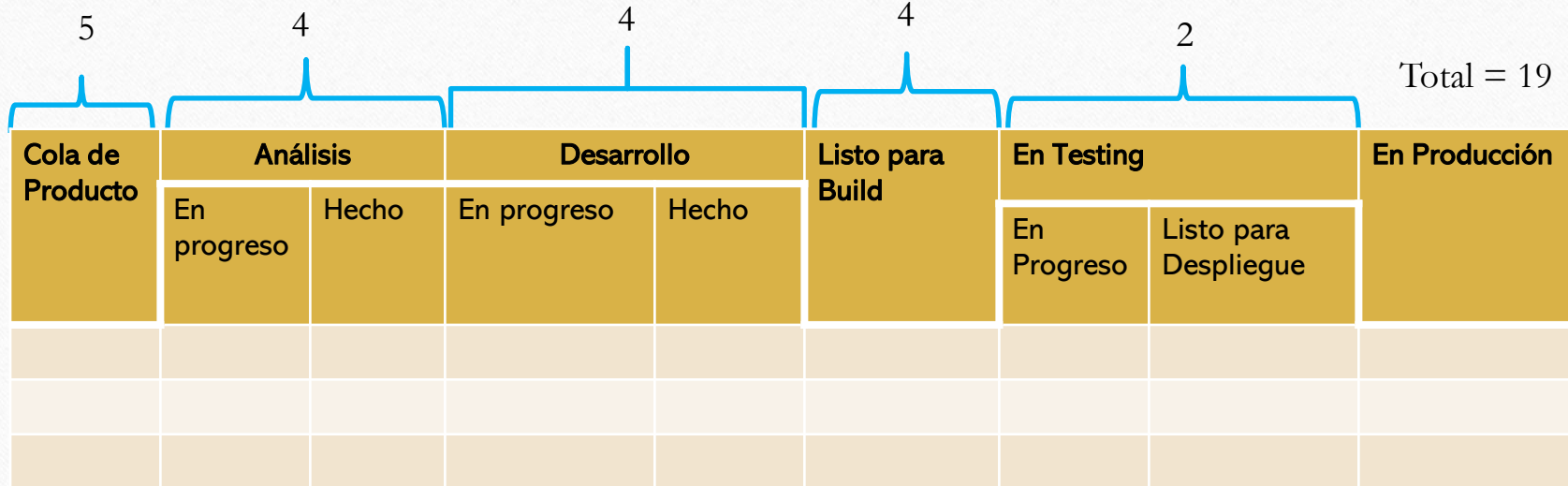
Desarrollo

- Mantenimiento
- Refactorización
- Actualización de Infraestructura

Solicitudes


- Solicitud de Cambio
- Sugerencias de Mejora

Definir el WIP...



Definir tipos de trabajo...

Asignando capacidad en función de la demanda

| Cola de Producto | Análisis | | Desarrollo | | Listo para Build | En Testing | | En Producción |
|--|-------------|-------|-------------|-------|------------------|-------------|-----------------------|---------------|
| | En progreso | Hecho | En progreso | Hecho | | En Progreso | Listo para Despliegue | |
|  Casos de Uso 60 % | | | | | | | | |
|  Mantenimiento 30 % | | | | | | | | |
|  Defectos 10% | | | | | | | | |

Políticas Explícitas para cada clase de servicio

Políticas para la clase de servicio expreso, un ejemplo



Expreso

Color de
tarjeta:
blanco

WIP = 1

Los demás
trabajos se
ponen en
espera

Se puede
exceder el
límite de
WIP para
procesar
este
trabajo

La capacidad
no se
reserva

De ser
necesario
se hace
una
entrega
especial,
para
ponerla
en
produc-
ción

Políticas Explícitas para cada clase de servicio

Políticas para la clase de servicio “Fecha Fija”, un ejemplo

Color de
tarjeta:
Rosa

Deben
adherirse al
WIP
definido

Fecha de
entrega en
la parte
superior

Permanecen
la cola hasta
que sea
conveniente
que ingresen

Si se retrasa
y la fecha de
entrega está
en riesgo
puede
promoverse a
la clase de
servicio
“expreso”

Son
entregados en
entregas
programadas
cuidando la
fecha de
entrega

**Fecha
Fija**

Políticas Explícitas para cada clase de servicio

Políticas para la clase de servicio “Estándar”, un ejemplo

Color de
tarjeta:
Amarillo

Deben
adherirse
al WIP
definido

Son
priorizados
y puestos
en la cola
con un
mecanismo
definido
basado en
valor de
negocio

Usan la
técnica FIFO,
si no hay
Expresos o
con Fecha
Fija

Pueden
analizarse
por tamaño,
en orden de
magnitud

Son
entregados
en entregas
programadas

Estándar

Políticas Explícitas para cada clase de servicio

Políticas para la clase de servicio “Intangible”, un ejemplo

Color de tarjeta: verde

Deben adherirse al WIP definido

Son priorizados y puestos en la cola con un mecanismo definido basado en valor de negocio

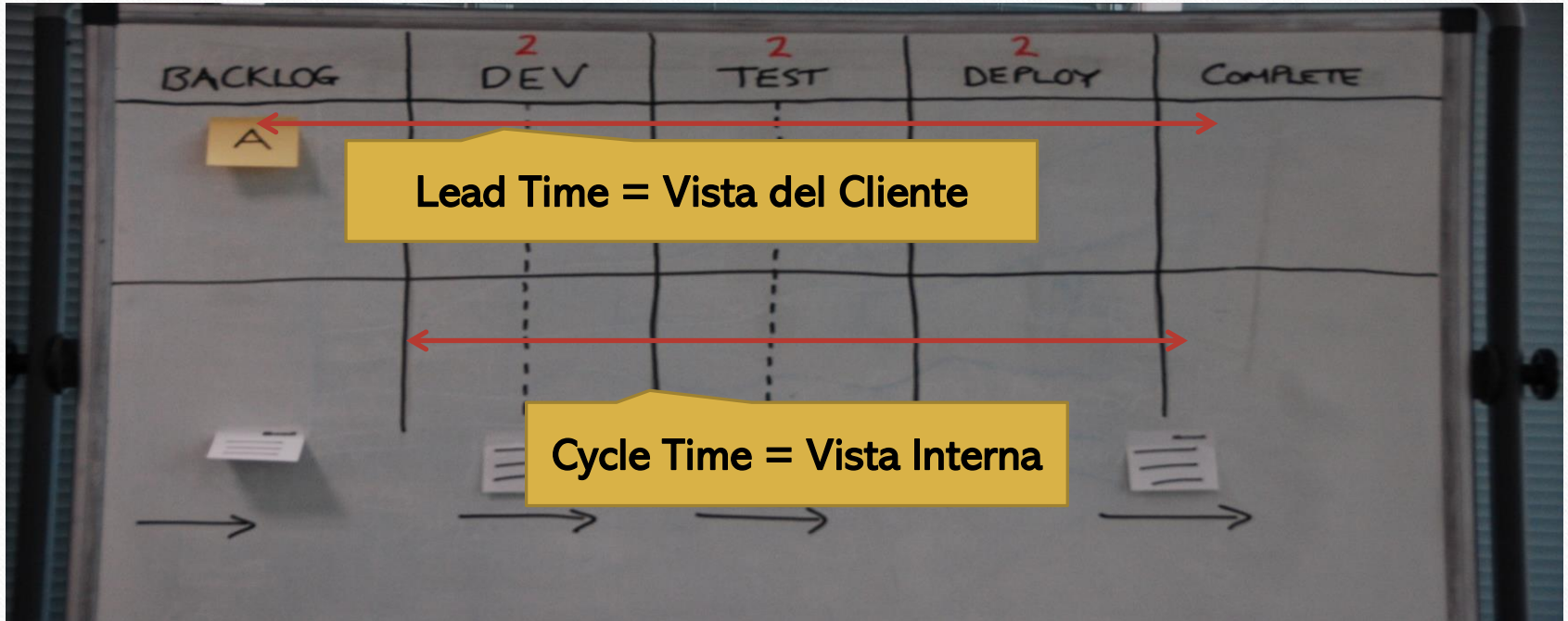
Usan la técnica FIFO, si no hay Expresos o con Fecha Fija

Pueden analizarse por tamaño, en orden de magnitud

Son entregados en entregas programadas

Intangible

KANBAN: Métricas Clave



KANBAN: Métricas Clave

Cycle Time (Tiempo de ciclo)

- Es la métrica que registra el tiempo que sucede entre el inicio y el final del proceso, para un ítem de trabajo dado. Se suele medir en días de trabajo o esfuerzo.
- Medición más mecánica de la capacidad del proceso
- **Ritmo de Terminación**

Lead Time (Tiempo de entrega)

- Es la métrica que registra el tiempo que sucede entre el momento en el cual se está pidiendo un ítem de trabajo y el momento de su entrega (el final del proceso). Se suele medir en días de trabajo.
- **Ritmo de entrega**

KANBAN: Métricas Clave

Touch Time (Tiempo de Tocado)

- El tiempo en el cual un ítem de trabajo fue realmente trabajado (o "tocado") por el equipo.
- Cuántos días hábiles pasó este ítem en columnas de "trabajo en curso", en oposición con columnas de cola / buffer y estado bloqueado o sin trabajo del equipo sobre el mismo.

$$\textit{Touch Time} \leq \textit{Cycle Time} \leq \textit{Lead Time}$$

Eficiencia del Ciclo de Proceso

$$\% \text{ Eficiencia ciclo proceso} = \textit{Touch Time} / \textit{Elapsed Time}.$$

