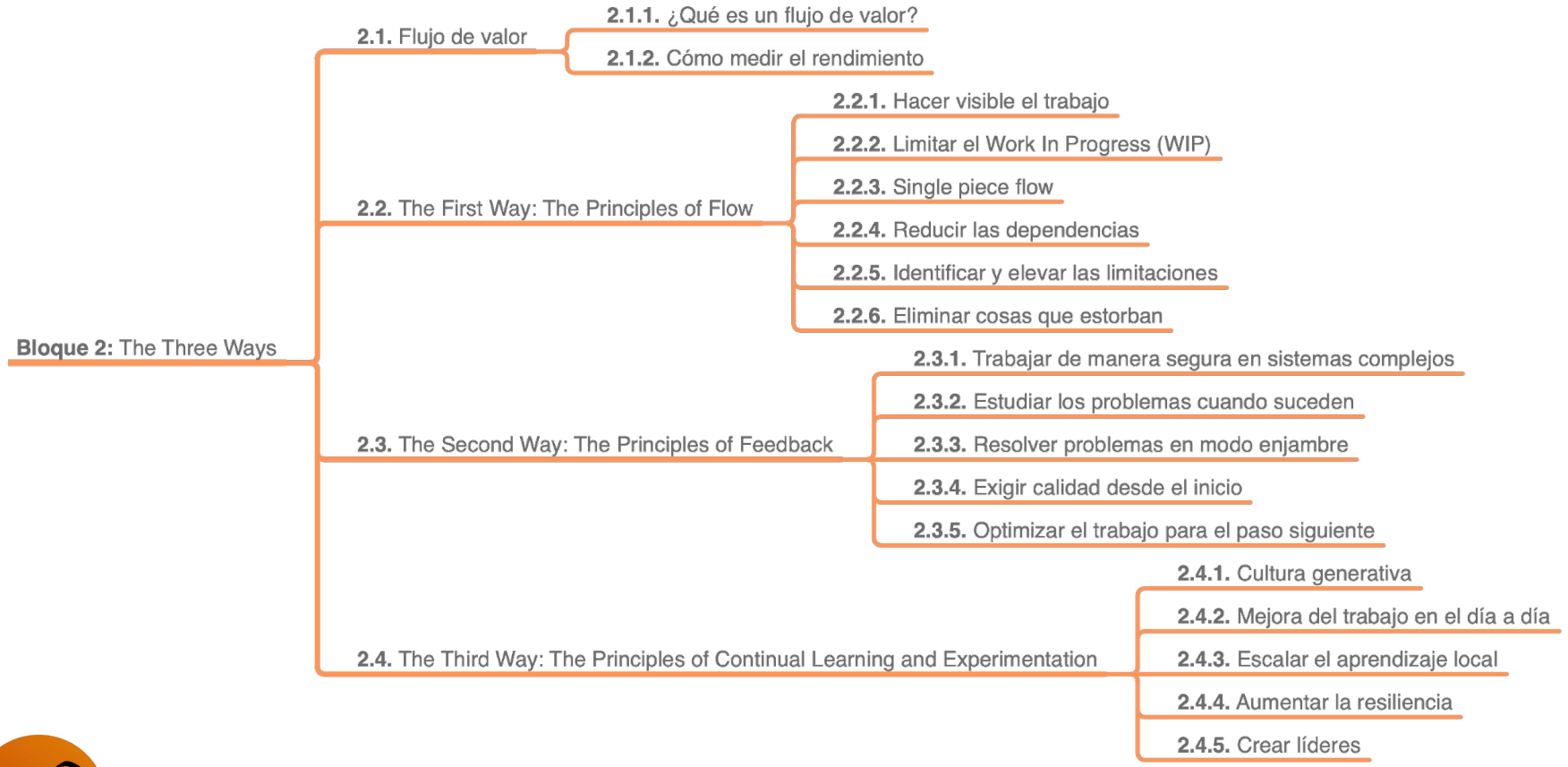
Bloque 2

The Three Ways



The Three Ways

The Three Ways hace referencia a una serie de principios sobre los que se sustenta **DevOps**.

Se dividen en tres grandes bloques.

The First Way

Flujo de trabajo

The First Way
Flujo de trabajo

The Second Way
Feedback

The First Way
Flujo de trabajo

The Second Way
Feedback

The Third Way
Experimentación

~ 2.1 ~

Flujo de valor

~ 2.1.1 ~

¿Qué es un flujo de valor?

Flujo de valor

Es la secuencia de actividades que una organización emprende para responder a una petición de un cliente. También puede verse como la secuencia de actividades necesarias para diseñar, producir y entregar un bien o un servicio a un cliente.

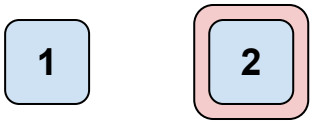
Flujo de valor

Un flujo de valor tecnológico es el proceso requerido para convertir una hipótesis de negocio en un servicio que aporta valor al cliente. Y esto solo lo hace cuando está en producción, por lo que debemos mimar todo el proceso para evitar caídas del servicio, problemas de seguridad, etc.

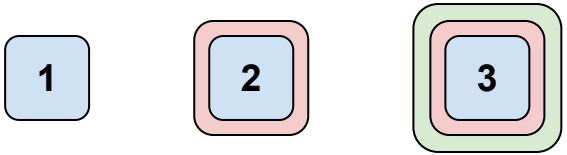
Flujo de valor

1

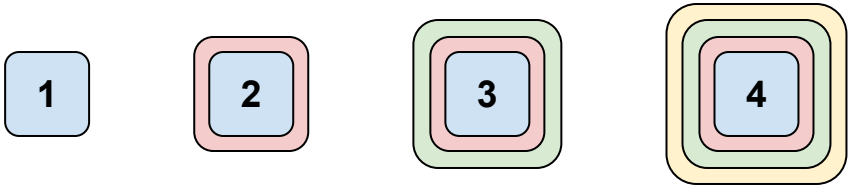
Flujo de valor



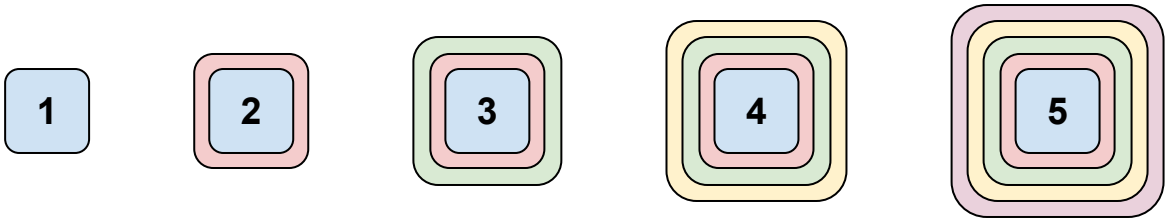
Flujo de valor



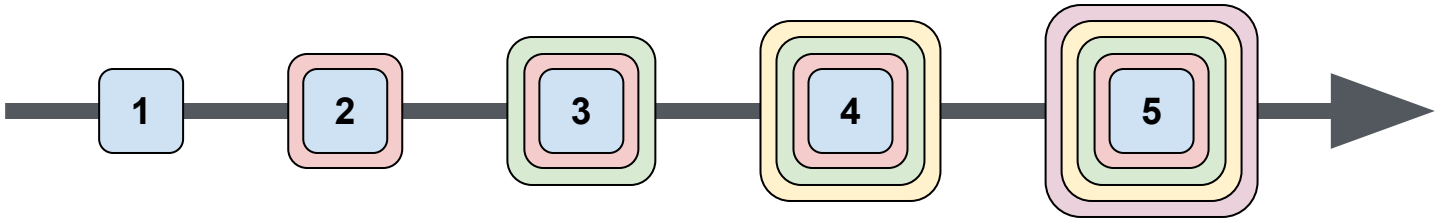
Flujo de valor



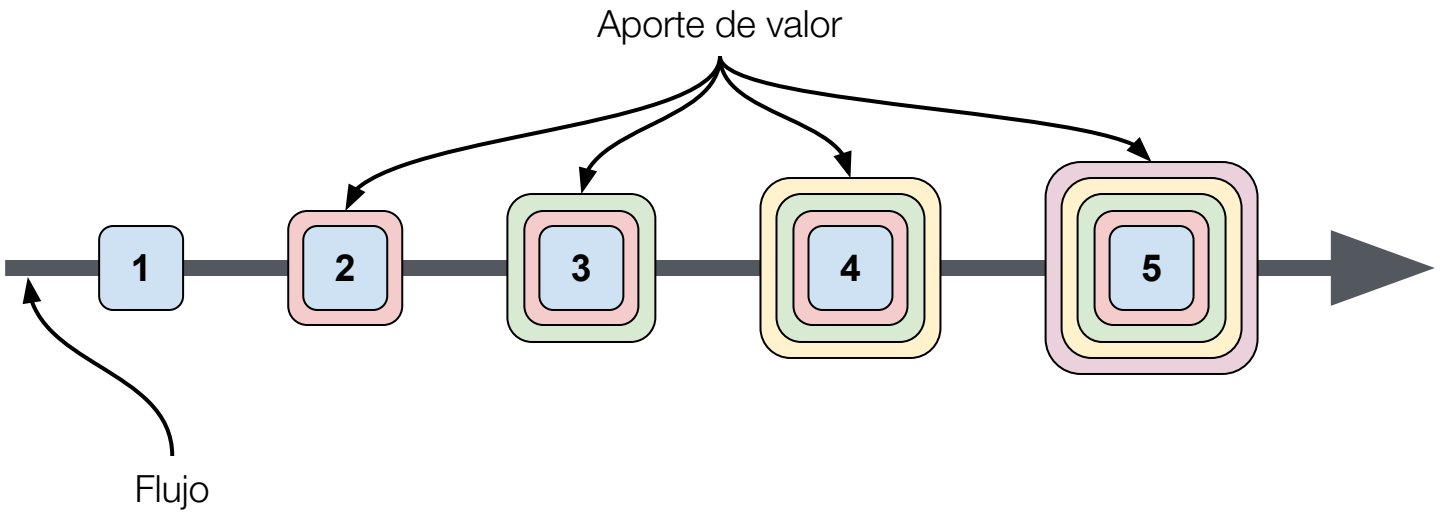
Flujo de valor



Flujo de valor



Flujo de valor



~ 2.1.2 ~

Cómo medir el rendimiento

Rendimiento

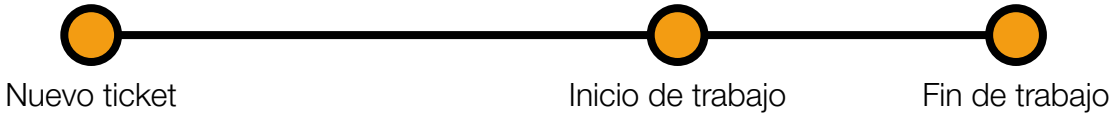
¿Cómo se mide el rendimiento en un flujo de valor?

- **Lead Time** → es el tiempo que transcurre desde que el cliente hace una petición hasta que obtiene valor de ella.
- **Process Time** → es el subconjunto del Lead Time que se inicia cuando la petición del cliente es atendida.

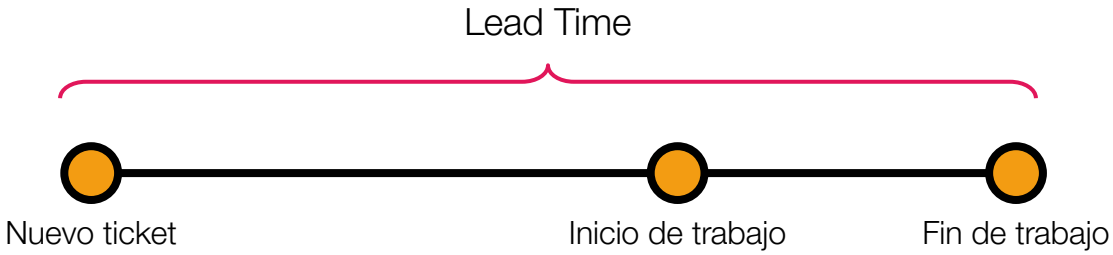
Rendimiento

- **%C/A** → indica cómo de completo y preciso es el trabajo recibido del paso anterior del flujo de valor. Es decir, si puede usarse tal cual o requiere retrabajo porque no se adapta a lo requerido.

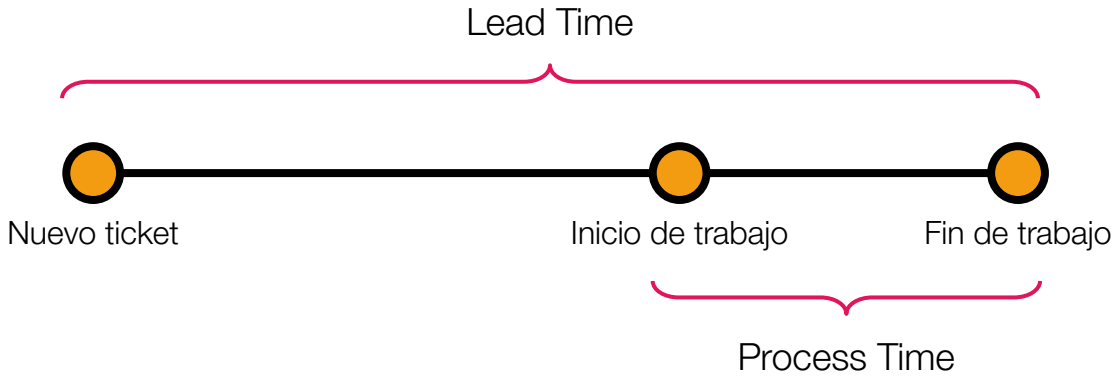
Rendimiento



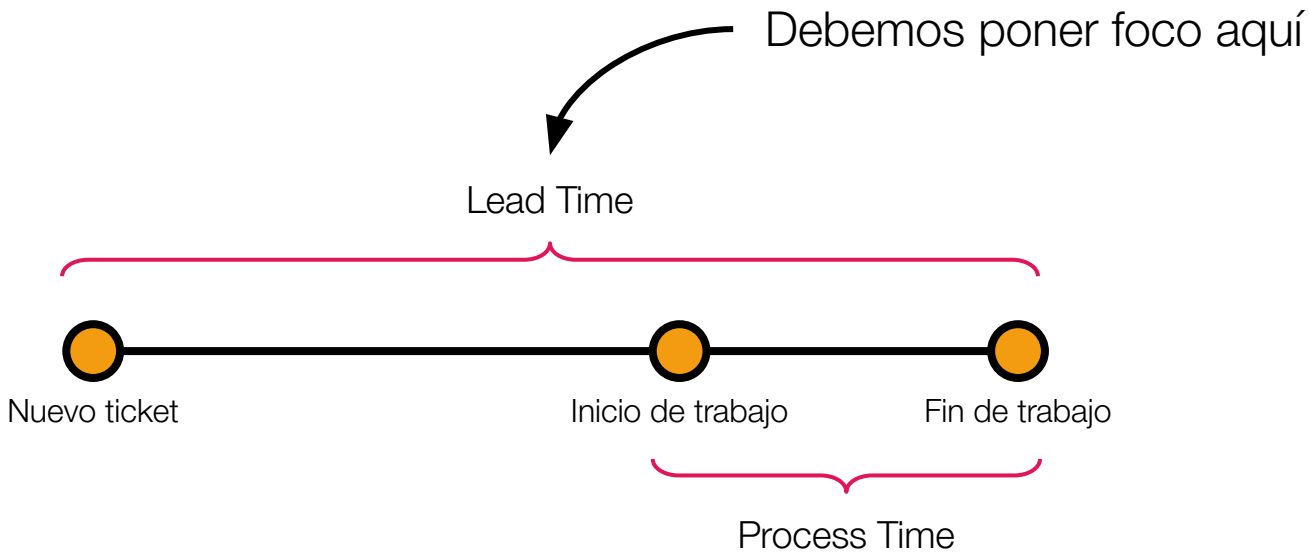
Rendimiento



Rendimiento



Rendimiento



~ 2.2 ~

The First Way

The Principles of Flow

The First Way
Flujo de trabajo

The Second Way
Feedback

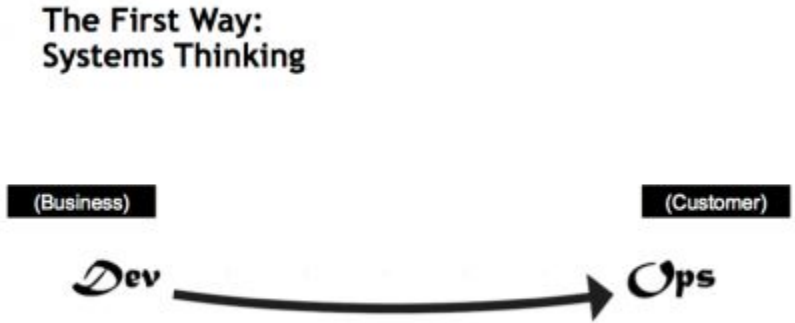
The Third Way
Experimentación

The First Way

Este First Way establece que:

- El trabajo debe fluir solo en una dirección.
- Ningún error conocido debe pasar al siguiente punto del flujo.
- Se debe buscar siempre la mejora e incremento del flujo.

The First Way



Site: IT Revolution | **Post:** The Three Ways: The Principles Underpinning DevOps | **Autor:** Gene Kim | **Enlace:** http://bit.ly/30GF1XL

The First Way

Los principios del First Way nos ayudan a entender el trabajo en IT como un flujo de valor, donde cada paso añade valor sobre el anterior.

¿Cuáles son los principios del First Way?

~ 2.2.1 ~

Hacer visible el trabajo

Visibilidad

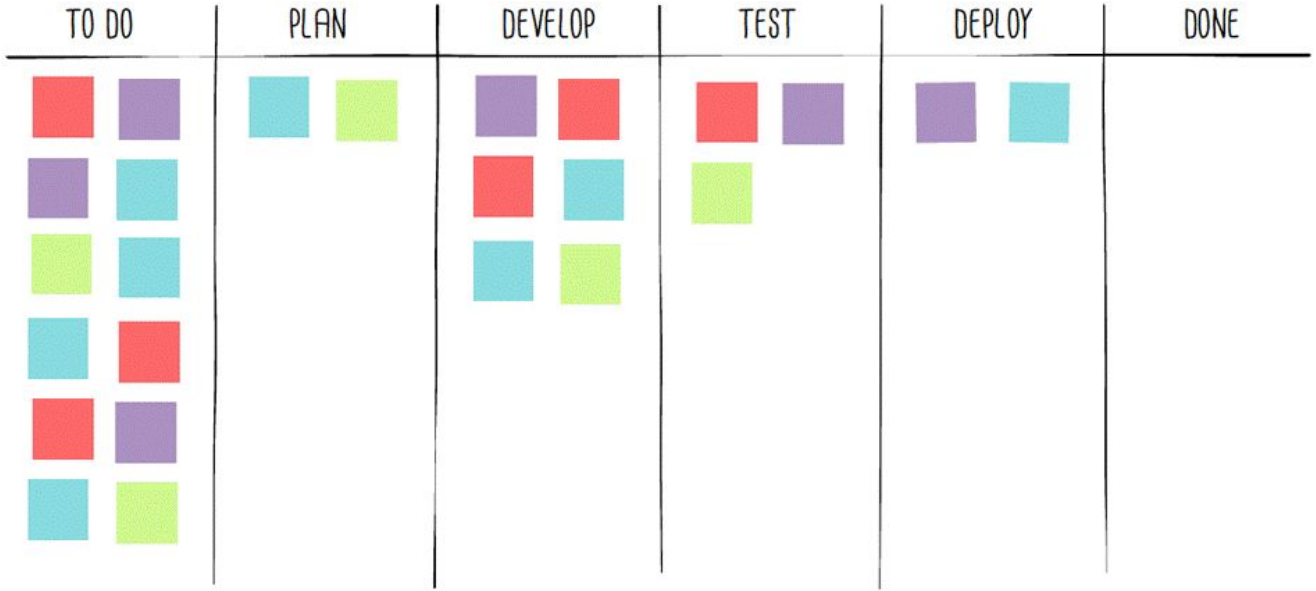
El trabajo en el sector IT es invisible, y por tanto complica detectar:

- Qué parte del todo puede estar frenando el flujo.
- Qué restricción o limitación puede estar acumulando trabajo.

Visibilidad

Podemos ayudarnos de paneles físicos o digitales para tener mayor visibilidad de aquello en lo que trabajamos, cuándo el trabajo fluye bien, y cuándo se encola y/o queda estancado.

Visibilidad



~ 2.2.2 ~

Limitar el Work In Progress (WIP)

WIP

El trabajo en el flujo de valor es dinámico. Es posible que se necesite satisfacer demandas de varios proyectos, provocando que:

- La prioridad del trabajo cambie, incluso a lo largo del día.
- El tiempo en completar las tareas se dilate más de lo debido.

WIP

Podemos limitar el WIP estableciendo límites a nivel de columna o de equipo de trabajo en un panel Kanban. Esto también ayuda a detectar bloqueos derivados de dependencias con otras tareas o equipos.

~ 2.2.3 ~

Single piece flow

Single piece flow

La producción en serie tiene los efectos siguientes:

- El nivel de Work In Progress (WIP) y variabilidad crecen.
- Los cambios que se llevan a producción son mayores, lo que dificulta la detección y corrección de errores.
- Aumenta de manera considerable el Lead Time.

Single piece flow

Site: Medium | **Post:** Single Piece Flow | **Autor:** Stefan Luyten | **Enlace:** http://bit.ly/2X4seMk

Single piece flow

Es importante hacer uso del single piece flow, ya que consigue reducir todo lo siguiente:

- Work In Progress (WIP).
- Lead Time.
- Tiempo invertido en la detección de errores.
- Tiempo invertido en retrabajo.

~ 2.2.4 ~

Reducir las dependencias

Dependencias

Pueden darse tiempos de espera de semanas/meses cuando:

- Gran cantidad de las operaciones requieren de otros equipos.
- La comunicación entre equipos no es óptima.
- El sistema de ticketing varía de un equipo a otro.
- Se pierde conocimiento y contexto en el traspaso de trabajo.

Dependencias

Se puede mitigar este problema reduciendo el número de traspasos, automatizando parte significativa del trabajo y reorganizando los equipos para puedan entregar valor de forma independiente y constante.

~ 2.2.5 ~

Identificar y elevar las limitaciones

Limitaciones

Debemos identificar las limitaciones de nuestro sistema para mejorar su rendimiento. Algunos ejemplos pueden ser los siguientes:

- **Creación de entorno** → para evitar largas esperas, se deben poder crear entornos de manera autónoma y bajo demanda.
- **Despliegue de código** → debe estar automatizado, de manera que cualquier desarrollador pueda hacer uso de ello.

Limitaciones

- **Configuración y ejecución de tests** → deben automatizarse para lanzar tests de manera segura, autónoma y en paralelo.
- **Arquitectura acoplada** → la arquitectura debe estar desacoplada y permitir que los cambios puedan hacerse de manera segura y con autonomía.

~ 2.2.6 ~

Eliminar cosas que estorban

Residuos

Los residuos suponen una amenaza para la viabilidad de un negocio, y por tanto hay que combatirlos y eliminarlos. Algunos ejemplos pueden ser los siguientes:

- **Trabajo parcialmente terminado** → el trabajo incompleto queda obsoleto y pierde valor con el paso del tiempo.

Residuos

- **Procesos demás** → cualquier trabajo realizado que no aporta valor alguno al cliente; aumenta el tiempo de entrega.
- **Features demás** → features que se añaden al servicio y que no son necesarios ni para el cliente ni para la organización; añaden complejidad y esfuerzos a la hora de testarlos.
- **Cambio de tareas** → estar en varios proyectos a la vez lleva a cambios de contexto continuos que suman esfuerzo y tiempo extra.

Residuos

- **Tiempos muertos** → retrasos que requieren la finalización del trabajo actual para continuar; aumentan el tiempo de entrega.
- **Movilidad** → esfuerzo necesario para mover información y materiales entre equipos.
- **Defectos** → todo tipo de información, materiales y/o productos que no son correctos, se han perdido o no son claros.

Residuos

- **Trabajo no estándar** → confianza en el trabajo manual o no estándar; toda dependencia con **Ops** debería estar automatizada y poder usarse de manera autónoma.
- **Heroicidades** → acciones no razonadas motivadas por malas decisiones sobre la consecución de objetivos de la organización.

~ 2.3 ~

The Second Way

The Principles of Feedback

The First Way
Flujo de trabajo

The Second Way
Feedback

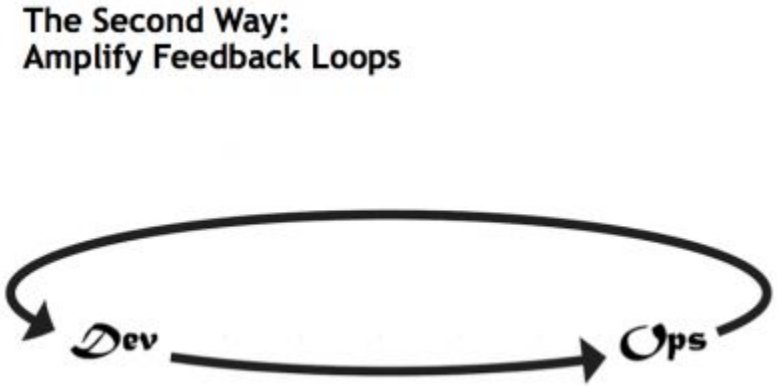
The Third Way
Experimentación

The Second Way

Este Second Way establece que:

- Se deben habilitar mecanismos de feedback continuo en sentido ascendente en el flujo de valor.
- El feedback debe ser rápido, frecuente y de gran calidad.
- Los problemas que tienen lugar deben entenderse como oportunidades de aprendizaje y mejora, es decir, ser constructivo.

The Second Way



Site: IT Revolution | **Post:** The Three Ways: The Principles Underpinning DevOps | **Autor:** Gene Kim | **Enlace:** http://bit.ly/30GF1XL

The Second Way

Los principios del Second Way nos ayudan a detectar y corregir errores de forma temprana, cuando aún no son problemas de gran magnitud. Además, facilitan la creación de aprendizaje que puede integrarse en trabajos futuros.

¿Cuáles son los principios del Second Way?

~ 2.3.1 ~

Trabajar de manera segura en sistemas complejos

Sistemas complejos

Un sistema complejo suele estar acoplado y su comportamiento no es predecible. Esto dificulta aislar los problemas cuando ocurren, y por tanto se propagan de la manera más imprevisible posible.

Sistemas complejos

Podemos mejorar la seguridad en un sistema complejo si se dan estas condiciones:

- El trabajo complejo debe gestionarse de forma que permita evidenciar problemas de diseño y operaciones.
- La resolución de problemas debe resultar en la rápida construcción de nuevos conocimientos.

Sistemas complejos

- Los nuevos conocimientos de ámbito local deben explotarse de manera global en la organización.
- Los líderes deben crear nuevos líderes que hagan crecer estas capacidades de manera continua.

~ 2.3.2 ~

Estudiar los problemas cuando suceden

Estudiar problemas

Se deben poner a pruebas las suposiciones de diseño y operaciones con el objetivo de incrementar el flujo de información en el sistema, desde y hacia el mayor número de áreas posibles. Cuantas más suposiciones se invaliden, más rápida será la detección y corrección de problemas.

Estudiar problemas

Los ciclos de feedback continuo, la automatización de integraciones y testeos, así como el uso de telemetría para evaluar el comportamiento de los servicios, son ejemplos de acciones que pueden ayudar.

~ 2.3.3 ~

Resolver problemas en modo enjambre

Enjambre

Cuando lo inesperado ocurre se debe acudir como un enjambre, movilizando a quien sea necesario para solucionarlo. El objetivo es contener el problema antes de que tenga la oportunidad de propagarse: se aborda y soluciona de manera inmediata.

- Previene la propagación del problema en el flujo de valor, donde el coste y esfuerzo para su reparación se incrementa exponencialmente y la deuda técnica se acumula.

Enjambre

- Previene que un equipo de trabajo comience una nueva tarea, la cual probablemente introducirá nuevos errores en el sistema.
- Previene la pérdida de información crítica, sobre todo en sistemas complejos donde los problemas ocurren de forma inesperada.
- Si el problema no es abordado, el equipo de trabajo puede tener el mismo problema en la siguiente operación, requiriendo más esfuerzo y trabajo en su arreglo.

~ 2.3.4 ~

Exigir calidad desde el inicio

Calidad

Responder de manera incorrecta a incidentes puede llevar a perpetuar modos de trabajo poco seguros. Añadir pasos de inspección y procesos de aprobación aumenta la probabilidad de problemas futuros.

Calidad

Ejemplos de mala calidad de control son:

- Necesitar de otro equipo para completar tareas tediosas, propensas a errores, y tareas manuales que pueden ser automatizadas.
- Necesitar aprobación de personas que están distantes del problema, forzándolas a tomar decisiones sin el conocimiento adecuado.

Calidad

- Crear un gran volumen de documentación de calidad y detalles cuestionables, que queda obsoleto rápidamente tras ser escrito.
- Enviar gran cantidad de trabajo a equipos y comités para su aprobación y procesamiento, quedando a la espera de respuestas.

Calidad

Todas las personas implicadas en un flujo de valor deben detectar problemas en sus respectivas áreas como parte del trabajo diario; esto añade responsabilidades de calidad y seguridad, así como una toma de decisiones donde corresponde.

La calidad de los sistemas se sustenta sobre la responsabilidad compartida de sus desarrolladores, que mejora resultados y acelera el aprendizaje.

~ 2.3.5 ~

Optimizar el trabajo para el paso siguiente

Optimizar trabajo

Lean establece dos tipos de cliente: externo, que paga por nuestros servicios, e interno, que es quien recibe y procesa nuestro trabajo inmediatamente después de liberarlo nosotros. El más importante es el siguiente en el flujo de valor.

Optimizar nuestro trabajo para nuestros clientes requiere empatía: al hacer esto creamos calidad en origen.

~ 2.4 ~

The Third Way

The Principles of Continual Learning and Experimentation

The First Way
Flujo de trabajo

The Second Way
Feedback

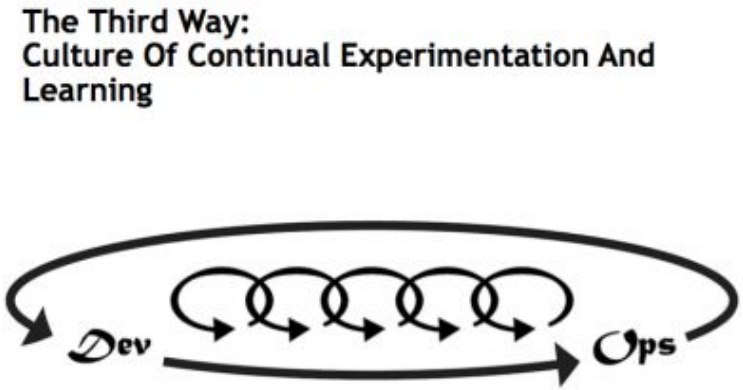
The Third Way
Experimentación

The Third Way

Este Third Way establece que:

- Se debe promover la experimentación y el aprendizaje.
- Se debe aprender tanto del éxito como del fracaso.
- Se debe alcanzar el dominio a través de la práctica.

The Third Way



Site: IT Revolution | **Post:** The Three Ways: The Principles Underpinning DevOps | **Autor:** Gene Kim | **Enlace:** http://bit.ly/30GF1XL

The Third Way

Los principios del Third Way abogan por una cultura de experimentación y mejora constante.

Tanto cultura como entorno son tan importantes como el propio trabajo que se está realizando.

¿Cuáles son los principios del Third Way?

~ 2.4.1 ~

Cultura generativa

Cultura generativa

Cuando un sistema falla, se tiende a pensar que la causa viene motivada por un error humano. Las capas de gestión pueden actuar culpando a la persona que causó el problema, e incluso insinuar castigos. Estas reacciones pueden llevar a una cultura del miedo en la que los problemas no se reportan, quedando ocultos hasta que ocurre una catástrofe.

Cultura generativa

Se identifican estos tipos de cultura:

- **Patológicas** → reina el miedo y las amenazas. Se acumula y retiene información por razones políticas, o se distorsiona para hacerla parecer mejor. El fracaso se oculta a menudo.

Cultura generativa

- **Burocráticas** → existen reglas y procesos que solo benefician a algunos. El fracaso es procesado por un sistema de juicio.
- **Generativas** → buscan y comparten de manera activa información que pueda ayudar a la organización a lograr su objetivo. Las responsabilidades son compartidas. Del fracaso se extrae reflexión.

Cultura generativa

Patológica	Burocrática	Generativa
La información se oculta	La información puede ser ignorada	La información se busca activamente
Se "dispara" a los mensajeros	Se tolera a los mensajeros	Se entrena a los mensajeros
Se eluden responsabilidades	Las responsabilidades se compartimentan	Las responsabilidades se comparten
Se desaconseja el trabajo con otros equipos	Se permite el trabajo con otros equipos, aunque se desaconseja	El trabajo con otros equipos es recompensado
Se oculta el fracaso	La organización es justa y misericordiosa	El fracaso causa indagación
Las nuevas ideas son aplastadas	Las nuevas ideas causan problemas	Las nuevas ideas son bienvenidas

Site: BMJ Journals | **Post:** A typology of organisational cultures | **Autor:** Ron Westrum | **Enlace:** http://bit.ly/2WotIUF

Cultura generativa

En nuestro flujo de valor debemos establecer las bases de una cultura generativa, esforzándonos en crear un sistema de trabajo seguro.

Cuando tiene lugar un problema, en vez de buscar culpables, pensamos cómo rediseñar el sistema para prevenir que vuelva a ocurrir. Por ejemplo, haciendo uso de post-mortems tras cada incidente.

~ 2.4.2 ~

Mejora del trabajo en el día a día

Mejora en el día a día

Los equipos no siempre pueden mejorar aquellos procesos en los que operan, resultando en problemas que no solo persisten en el tiempo, sino que empeoran.

En un flujo de valor tecnológico, la ausencia de soluciones se materializa en deuda técnica; esta se acumula a tal nivel que los equipos tienen que dar rodeos para evitar males mayores, haciendo que los ciclos de trabajo sean improductivos.

Mejora en el día a día

Debemos reservar tiempo de manera deliberada para pagar deuda técnica, arreglar defectos, y refactorizar y mejorar áreas problemáticas de nuestro código y entornos.

Se puede reservar tiempo en cada ciclo de desarrollo, o bien programando eventos *Kaizen Blitz*, que son periodos de tiempo en los que los ingenieros se auto organizan en equipos para resolver problemas concretos.

~ 2.4.3 ~

Escalar el aprendizaje local

Escalar aprendizaje

Deben existir mecanismos para escalar a toda la organización los nuevos aprendizajes que se dan de manera local, de manera que todos puedan beneficiarse de este conocimiento.

Esto asegura que el trabajo pueda afrontarse teniendo acceso a la información y al conocimiento acumulado de la experiencia colectiva en trabajos similares.

Escalar aprendizaje

En un flujo de valor tecnológico esto puede lograrse:

- Haciendo que los informes de post-mortem generados estén accesibles para su consulta por todos los equipos.
- Habilitando repositorios que permitan compartir con toda la organización el código, las librerías y los artefactos generados.

~ 2.4.4 ~

Aumentar la resiliencia

Resiliencia

Las organizaciones viven en continuo estado de cambio y tensión, y es importante aprovechar estos momentos para mejorar nuestra resiliencia.

Esto fue apodado por Nassim Nicholas Taleb como antifragilidad.

Resiliencia

Ejemplos para potenciar la antifragilidad y, por tanto, la resiliencia:

- Incrementar la cobertura de los tests.
- Replantear parte o toda la arquitectura de una solución.
- Poner en práctica fallos a gran escala de manera controlada.

~ 2.4.5 ~

Crear líderes

Líderes

La labor de un líder es provocar las condiciones necesarias para que su equipo consiga la excelencia en el trabajo del día a día. Crear esta excelencia requiere de líderes y seguidores, entre los cuales se da una situación de dependencia y respeto mutuo.

Líderes

Ninguno de estos perfiles funciona en solitario:

- El líder no suele estar lo suficientemente cerca del trabajo de más bajo nivel, por lo que no puede ayudar de manera directa a solucionar los problemas.
- Los seguidores, trabajadores de primera línea, no suelen tener contexto a nivel de organización, ni tampoco autoridad para acometer cambios más allá de su área de trabajo.

Líderes

Es responsabilidad del líder potenciar el valor del aprendizaje y de la resolución de problemas de forma disciplinada. Debe guiar a la persona en la resolución del problema.