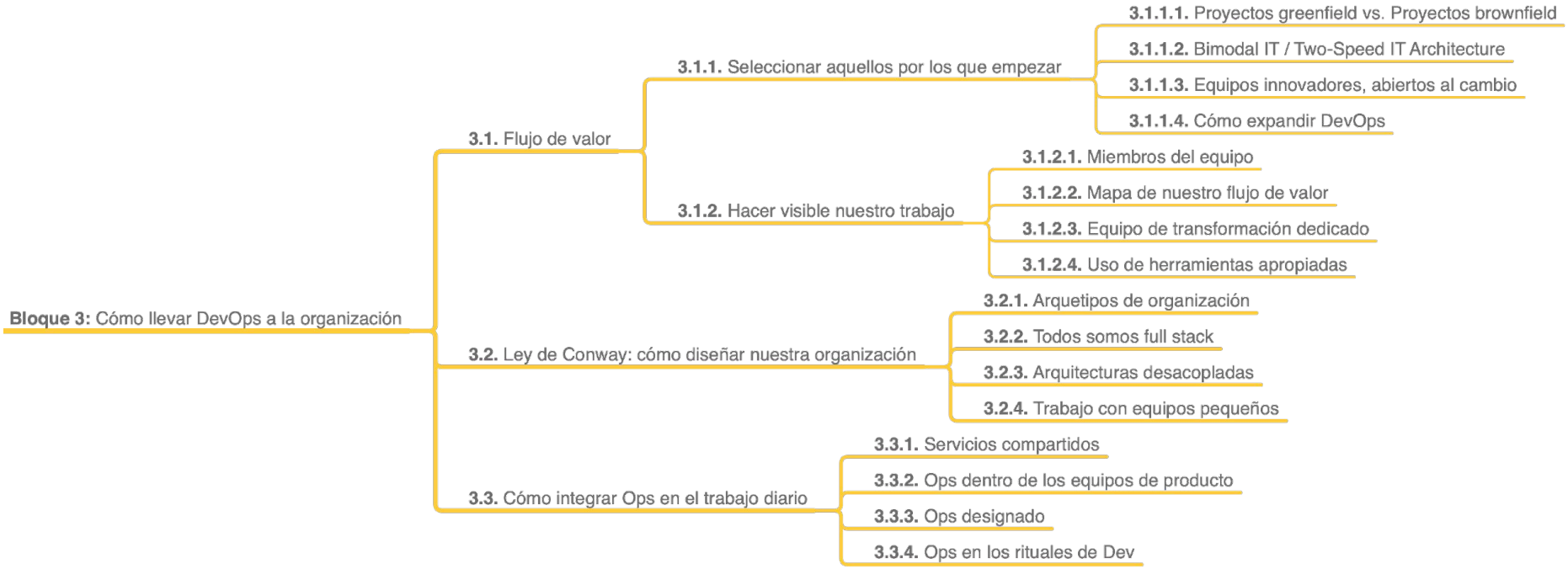
Bloque 3

Cómo llevar DevOps a la organización



~ 3.1 ~

Flujo de valor

~ 3.1.1 ~

Seleccionar aquellos por los que empezar

~ 3.1.1.1 ~

Proyectos greenfield vs. Proyectos brownfield

Tipo de proyecto

Un proyecto greenfield es aquel que:

- Se desarrolla desde desde cero.
- No tienen integración con sistemas heredados.
- Carece de restricciones impuestas por un algún trabajo anterior.

Tipo de proyecto

Un proyecto brownfield es aquel que:

- Supone trabajar sobre un producto o servicio existente.
- Tiene restricciones debido a dependencias con otros sistemas.
- Acumula deuda técnica.

Tipo de proyecto

Greenfield

Brownfield

~ 3.1.1.2 ~

Bimodal IT / Two-Speed IT Architecture

Bimodal IT

Bimodal IT es un término definido por Gartner que recoge los distintos modos en que las organizaciones deben gestionar y controlar sus proyectos. McKinsey define una aproximación similar llamada Two-Speed IT Architecture.

Los Modos 1 y 2 de Bimodal IT se caracterizan por lo siguiente:

Bimodal IT

El Modo 1 de trabajo:

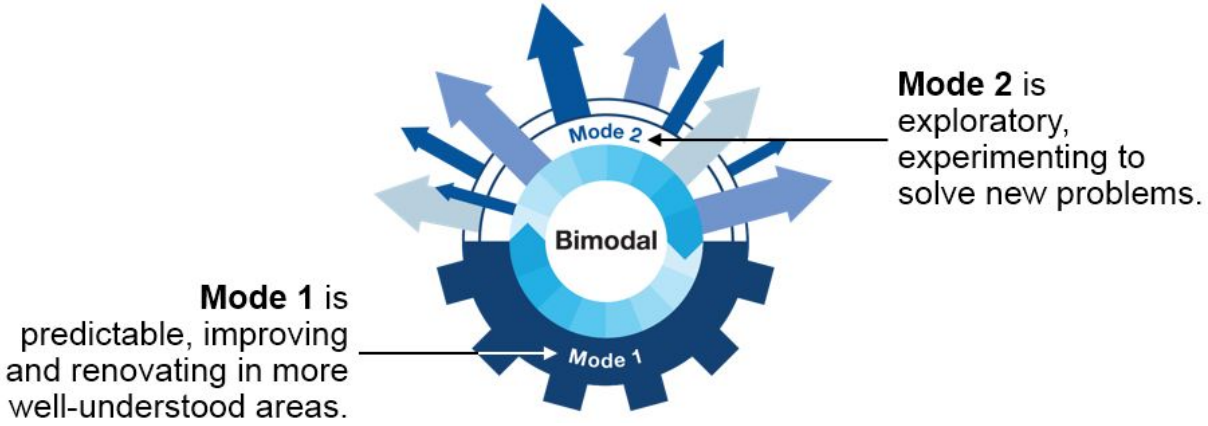
- Prioriza la estabilidad, la fiabilidad y la robustez.
- Pensado para áreas y sistemas predecibles.
- Promueve el uso de metodologías de gestión tradicionales.
- Busca que la organización ponga foco en *doing it right*.

Bimodal IT

El Modo 2 de trabajo:

- Prioriza la innovación y la entrega rápida.
- Pensado para áreas y sistemas con cierto grado de incertidumbre.
- Promueve el uso de metodologías ágiles de gestión.
- Busca que la organización ponga foco en *doing it fast*.

Bimodal IT



Site: Gartner | **Post:** Gartner IT Glossary - Bimodal | **Enlace:** https://gtnr.it/2Kym2J2

~ 3.1.1.3 ~

Equipos innovadores, abiertos al cambio

Equipos innovadores

En todas las organizaciones hay equipos y personas que afrontan las nuevas ideas con distinta actitud. Los *Innovators* y *Early Adopters* muestran buena disposición, mientras que los grupos más conservadores se resisten al cambio.

Equipos innovadores

Innovators	Techies → Son agentes del cambio
Early adopters	Visionarios → Van un paso por delante de la mayoría
Early majority	Pragmáticos → Desinteresados, se dejan llevar por la masa
Late majority	Conservadores → Solo participan en aquello que está probado
Laggards	Escépticos → Adalides del "si funciona, no lo toques"

Equipos innovadores

Laggard

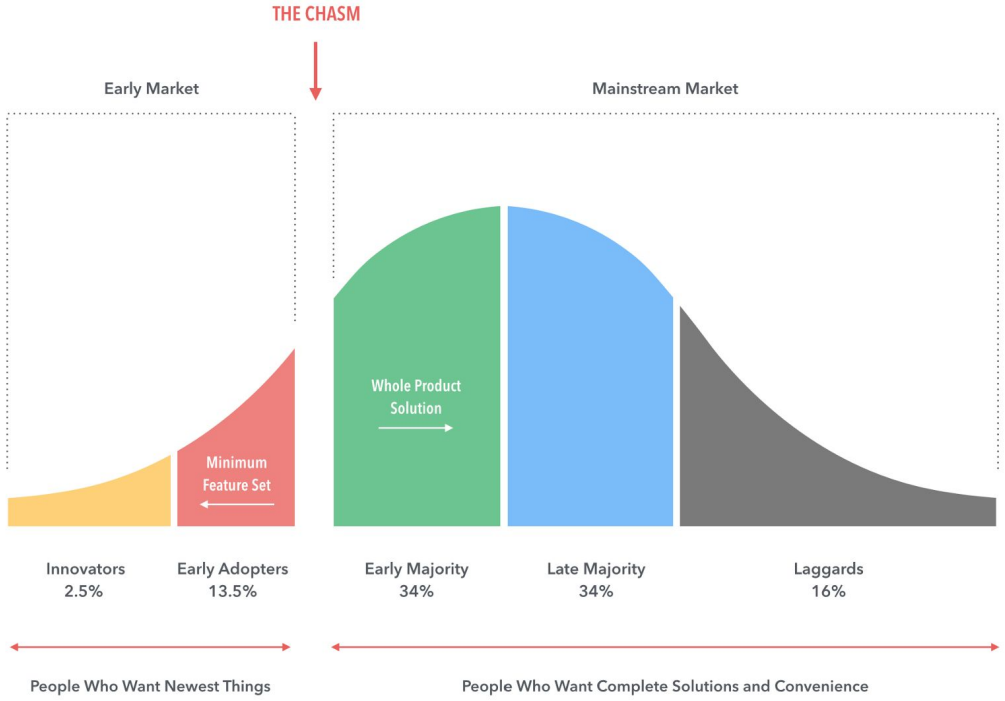
Early Adopter

Equipos innovadores

Nuestro objetivo es dar con esos equipos que creen en la necesidad de implantar los principios y prácticas de **DevOps**, y que demuestran tener habilidades para innovar y mejorar sus procesos. Una vez localizados, ponemos foco en conseguir pequeños logros que poco a poco vayan generando una buena base.

Nota: Hacer *big bang approach* está desaconsejado.

Equipos innovadores



Site: Prototypr | **Post:** Design for “Crossing the Chasm” | **Autor:** Shah Mohammed | **Enlace:** http://bit.ly/31dSiak

~ 3.1.1.4 ~

Cómo expandir DevOps

Expandiendo DevOps

Las fases que idealmente deben llevar a cabo los agentes del cambio para expandir **DevOps** en la organización son las siguientes:

- **Encontrar Innovators y Early Adopters** → se pone esfuerzo en aquellos equipos que quieren ayudar. Es deseable que alguno de los miembros tenga influencia a nivel de organización y pueda aportar credibilidad a la iniciativa.

Expandiendo DevOps

- **Construir masa crítica y mayoría silenciosa** → se expanden las prácticas **DevOps** a otros equipos y flujos de valor para asentar una base. Es importante evitar batallas políticas que puedan arruinar la iniciativa.
- **Identificar los detractores** → son personas con influencia que pueden sabotear la iniciativa. Es necesario conseguir logros suficientes para proteger la iniciativa y placar a los detractores.

~ Resumen 3.1.1 ~

Cualquier proyecto, greenfield o brownfield, es una opción.

Identificar agentes del cambio que ayuden desde abajo.

Construir masa crítica a base de pequeños logros.

~ 3.1.2 ~

Hacer visible nuestro trabajo

~ 3.1.2.1 ~

Miembros del equipo

Equipo normal

Una vez seleccionada la aplicación o el servicio en el que aplicar nuestra iniciativa **DevOps**, debemos identificar los miembros que participarán en el flujo de valor. Generalmente se incluye a:

- **Product Owner** → persona de negocio que define los siguientes pasos que deben darse en la construcción del servicio.
- **Development** → equipo responsable de desarrollar las nuevas funcionalidades del servicio.

Equipo avanzado

- **QA** → equipo encargado de que los ciclos de feedback tengan lugar para asegurar que el servicio funciona tal y como se espera.
- **Operations** → equipo responsable del mantenimiento del entorno de producción, así como de asegurar los niveles exigidos para este.
- **Infosec** → equipo encargado de la seguridad, tanto a nivel de sistemas como de datos.

Equipo equipo de ricos

- **Release managers** → personas encargadas de gestionar y coordinar los procesos de despliegue a producción.
- **Value stream manager** → persona responsable de asegurar que se cumplan los requisitos definidos para el flujo de valor, tanto aquellos exigidos por el cliente como los establecidos por la organización.

~ 3.1.2.2 ~

Mapa de nuestro flujo de valor

Mapa del flujo de valor

Es necesario documentar cómo el trabajo se lleva a cabo. Para ello, se propone un workshop con todos los miembros del flujo de valor, donde se pone foco en detectar las siguientes áreas:

- Aquellas en las que el trabajo sufre tiempos de espera, como procesos de aprobación, revisiones de seguridad, etc.
- Aquellas que generan y/o reciben una gran cantidad de retrabajo.

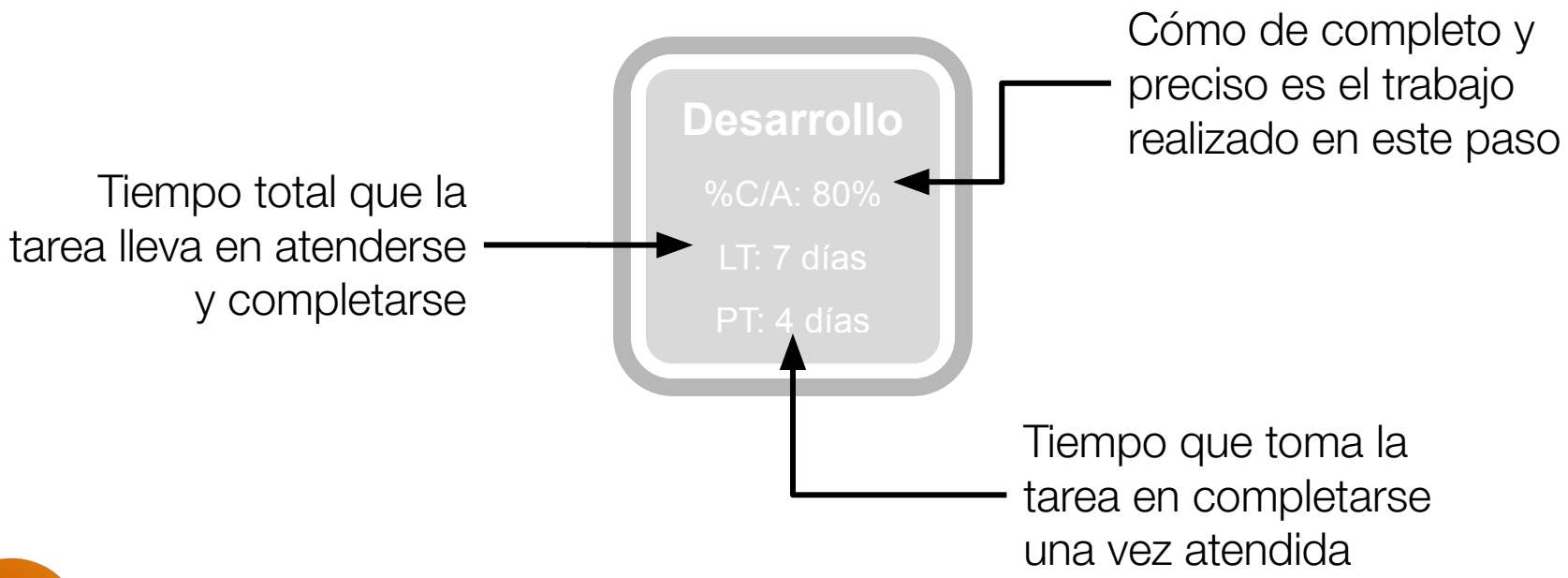
Mapa del flujo de valor

Se documenta en un diagrama de alto nivel, representando con bloques los distintos pasos del flujo de valor. Cada uno de estos bloques debe incluir las métricas siguientes que ya conocemos:

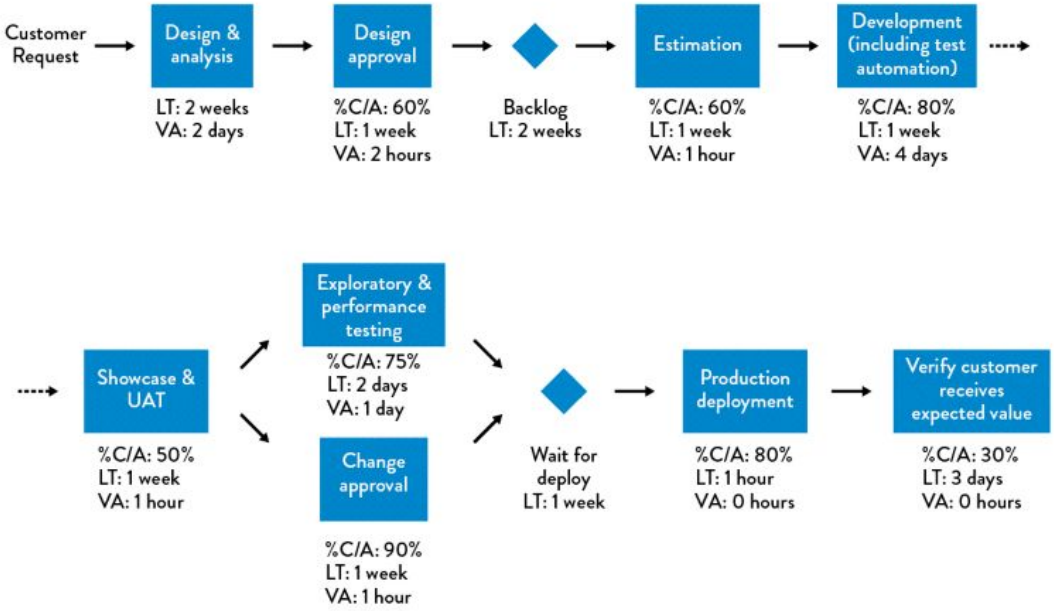
- Lead Time
- Process Time
- %C/A

Mapa del flujo de valor

Mapa del flujo de valor



Mapa del flujo de valor



Aggregate values:
Total lead time: 10 weeks
Value added time: 7.5 days
Percent complete and accurate: 8.6%

Site: Caylent | **Post:** DevOps Handbook Series Part 2 | **Autor:** Stefan Thorpe | **Enlace:** http://bit.ly/2K8inSW

Mapa del flujo de valor

Una vez se dispone del diagrama completo, se estudian las métricas para decidir cuál de ellas guiará los esfuerzos de mejora.

Es aconsejable definir un diagrama alternativo que represente el punto ideal al que se quiere llegar en los siguientes meses de trabajo.

Mapa del flujo de valor

Actual

Ideal

~ 3.1.2.3 ~

Equipo de transformación dedicado

Equipo de transformación

Existe la posibilidad de crear un equipo de transformación dedicado que pueda operar al margen del resto de la organización. Su cometido es conseguir resultados claros y medibles a nivel de sistema. Por ejemplo:

- Asegurar que el tiempo en llevar a producción un código subido al repositorio no excede a una semana en el 95% de los casos.
- Asegurar que las liberaciones de código a producción se hagan en horario de oficina sin corte alguno del servicio en cuestión.

Equipo de transformación

Montar este equipo consiste en:

- Asignar miembros con dedicación exclusiva a esta tarea.
- Asignar miembros con habilidades en distintos dominios.
- Asignar miembros asentados en la organización que gocen de buena relación con ésta y de respeto mútuo.
- Crear un espacio físico aislado del resto para este equipo.

Equipo de transformación

El equipo, una vez montado, tiene el siguiente cometido:

- Acordar un objetivo compartido con la organización que aporte valor a ésta y a los clientes. Debe ser medible y tener un deadline definido.
- Mantener planificaciones e iteraciones manejables en tiempo; esto permite ser flexible y poder repriorizar y replanificar rápidamente.

Equipo de transformación

- Reservar al menos un 20% del tiempo de cada ciclo para tareas no funcionales y reducción de deuda técnica.
- Hacer visible el estado del trabajo para evaluar el progreso.

~ 3.1.2.4 ~

Uso de herramientas apropiadas

Herramientas

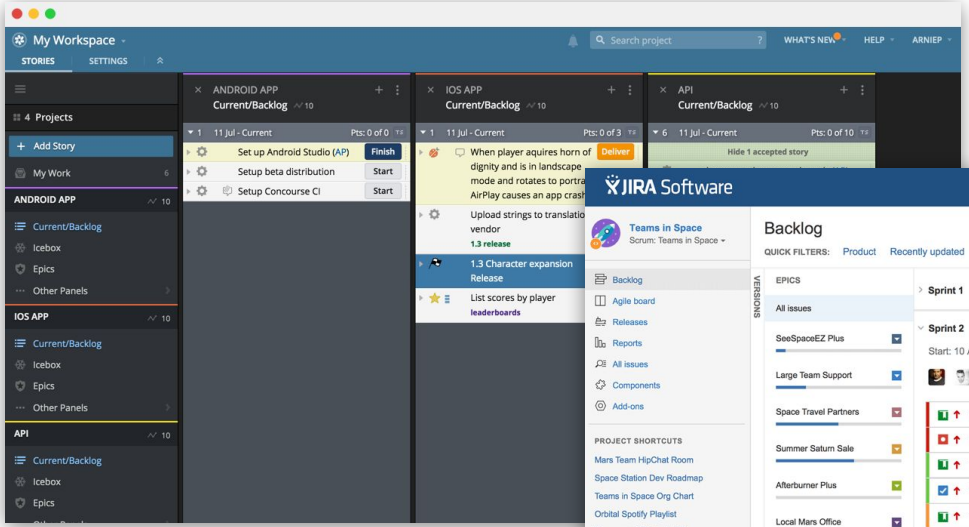
Dev y **Ops** comparten objetivos, backlog de trabajo y vocabulario, por tanto, deben hacer uso de herramientas que potencien la cultura. Esto permite:

- Tener un backlog unificado donde todos puedan priorizar mejoras del proyecto desde una perspectiva global, seleccionando aquellos que más valor aportan o mayor deuda técnica reducen.

Herramientas

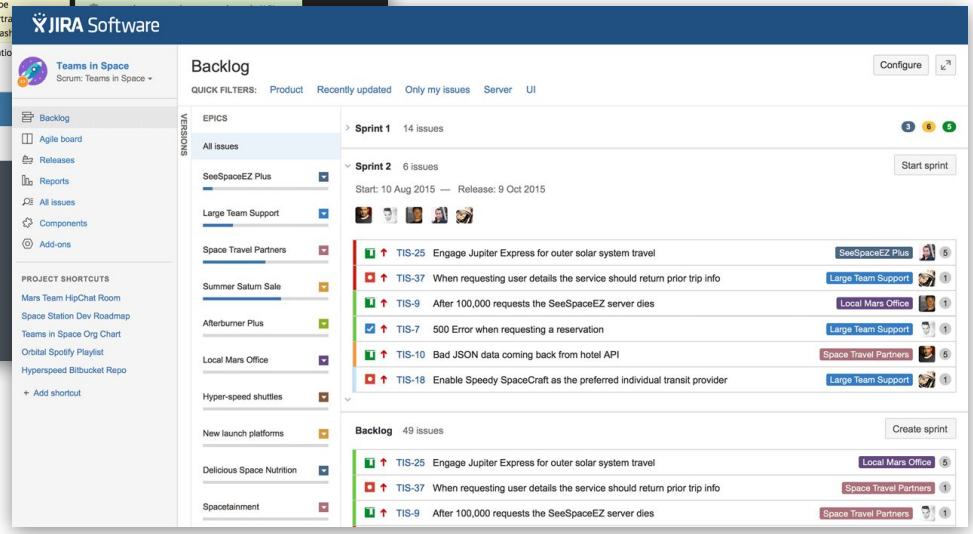
- Compartir información rápidamente y mantenerla en el tiempo, de manera que pueda analizarse a futuro en sesiones de post-mortem.

Herramientas

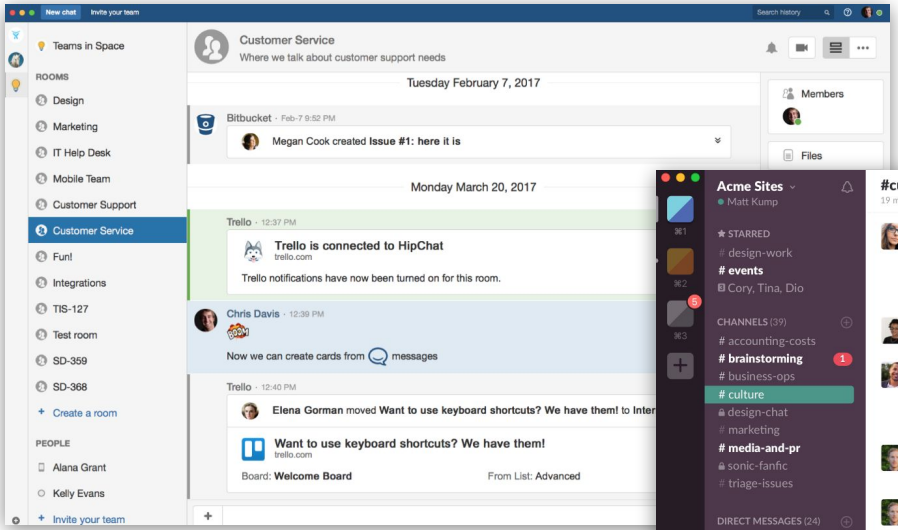


Pivotal Tracker

Jira

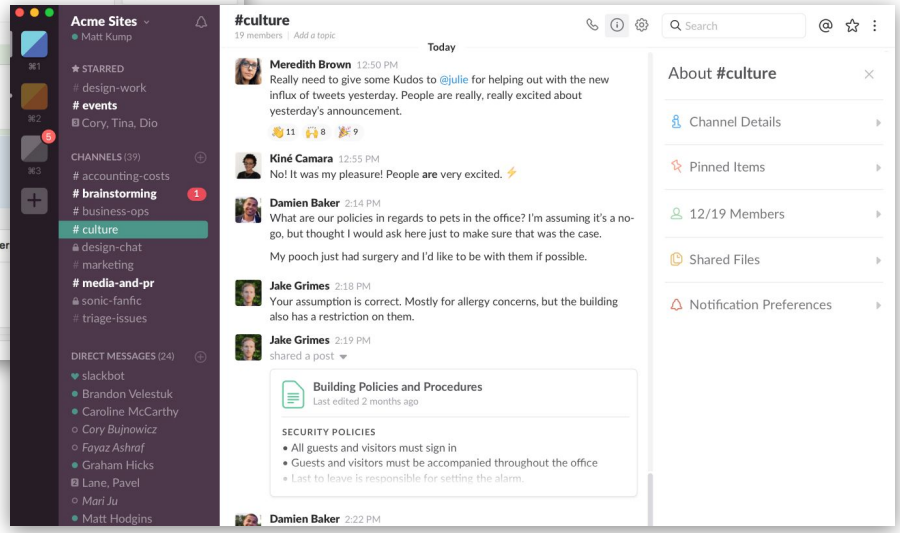


Herramientas



Hipchat

Slack



~ Resumen 3.1.2 ~

Seleccionar los miembros que formarán parte del equipo.

Identificar problemas del flujo de valor con la ayuda de un mapa.

Establecer un equipo de transformación dedicado.

Hacer uso de herramientas comunes que ayuden en el día a día.

~ 3.2 ~

Ley de Conway: cómo diseñar nuestra organización

Ley de Conway

La ley de Melvin Conway dice lo siguiente:

> *Las organizaciones dedicadas al diseño de sistemas [...] están abocadas a producir diseños que son copias de las estructuras de comunicación de dichas organizaciones.*

Ley de Conway

Que fue refinada más adelante por Eric S. Raymond:

> *La organización del software y la organización del equipo que lo produce son congruentes.*

~ 3.2.1 ~

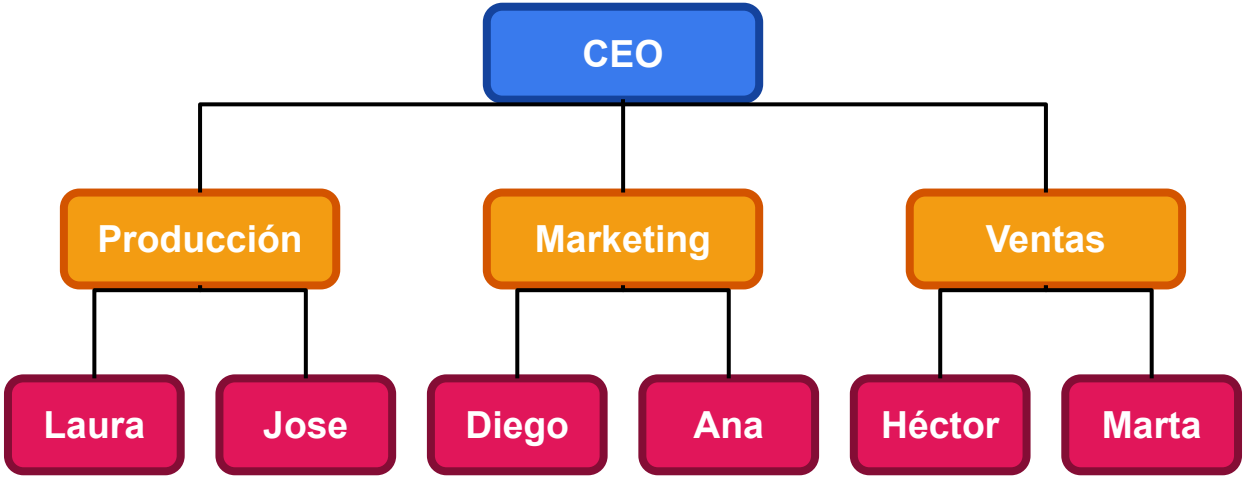
Arquetipos de organización

Arquetipos

Son tres los principales tipos de estructura organizacional:

- **Functional-oriented** → están optimizadas para la creación de *expertise*, la división del trabajo y la reducción de costes. Suelen tener jerarquías muy verticales.

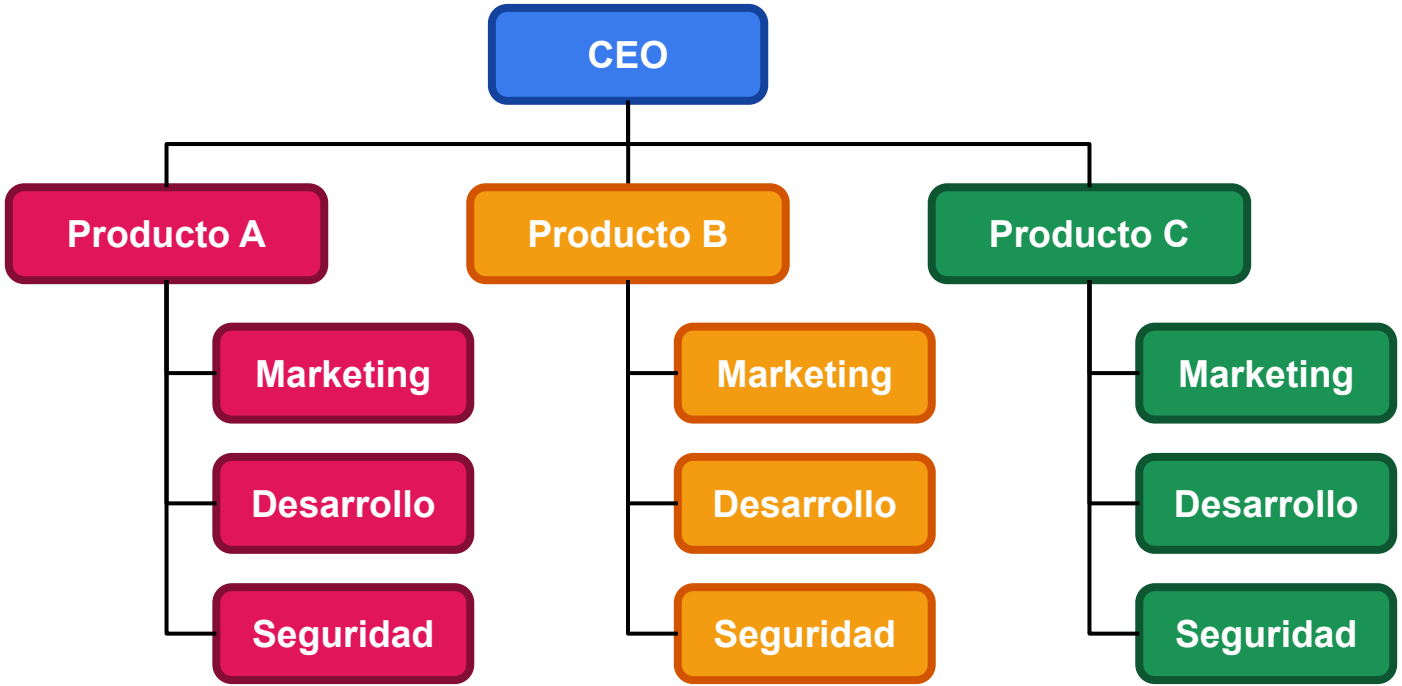
Arquetipos



Arquetipos

- **Market-oriented** → están optimizadas para responder rápidamente a las necesidades de los clientes. Tienden a ser planas, compuestas por múltiples disciplinas que actúan de manera *cross*.

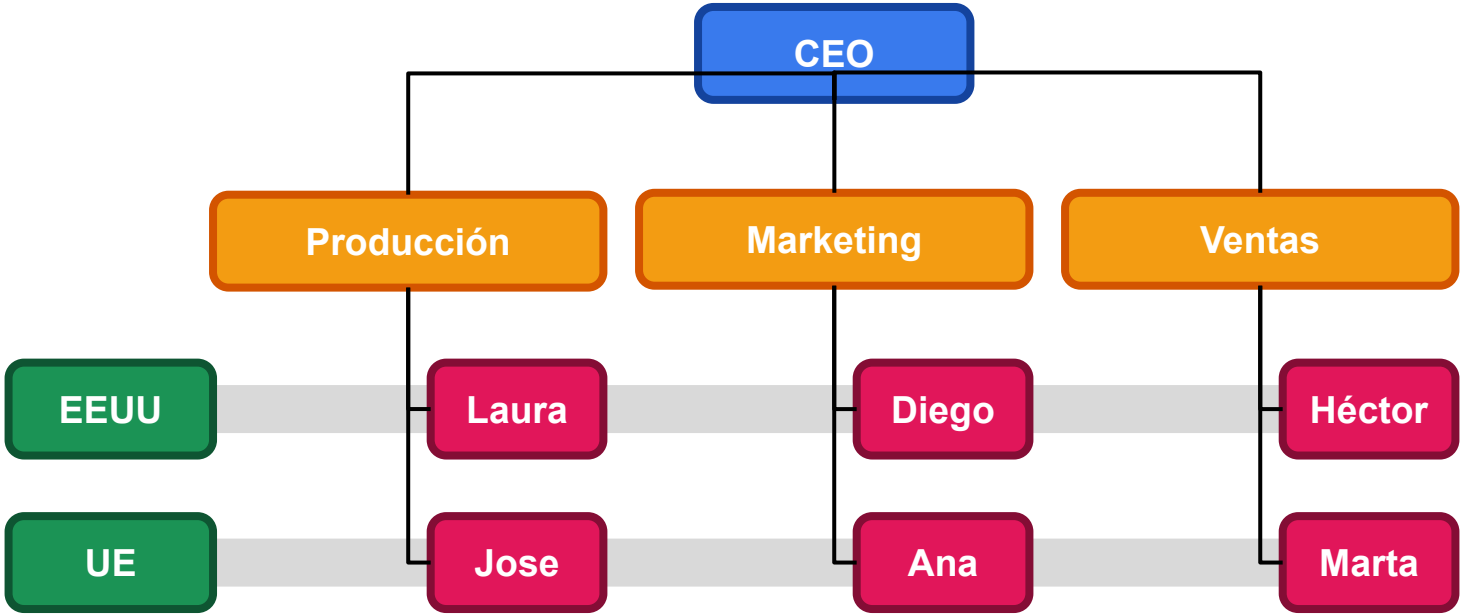
Arquetipos



Arquetipos

- **Matrix-oriented** → intentan combinar lo mejor de las estructuras *Functional-oriented* y *Market-oriented*. Son estructuras complejas que incluso llegan a complicar la consecución de objetivos, tanto los funcionales como los de mercado.

Arquetipos



~ 3.2.2 ~

Todos somos full stack

Full stack

Cuando los departamentos se especializan en exceso se convierten en silos; cualquier actividad operacional requiere de traspasos y colas de espera entre estos departamentos, lo que lleva a un aumento del Lead Time.

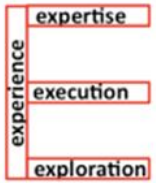
Full stack

Cada día nos apoyamos en nuevas tecnologías y necesitamos ingenieros especializados en aquellas que usamos. No interesa crear especialistas, sino generalistas que estén familiarizados con todo el stack tecnológico de los servicios o aplicaciones en los que trabajan.

Full stack

Creamos un *full stack* dando oportunidades a los ingenieros para aprender todas las habilidades necesarias para construir y ejecutar los sistemas de los que son responsables y, de manera periódica, haciendo que roten a través de diferentes roles.

Full stack

		
Very deep expertise in one area. Very few skills or experience in other areas.	Very deep expertise in one area. Broad skills across many areas.	Very deep expertise in a few areas. Experience across many areas. Proven execution skills. Always innovating.
Creates a bottleneck quickly. Insensitive to downstream waste and impact. Prevents planning flexibility or absorption of variability.	Can step up to remove bottlenecks. Sensitive to downstream waste and impact. Helps make planning flexible and absorbs variability.	Almost limitless potential.

Site: The Chief Curiosity Officer Blog | **Post:** Business Trend: “E-Shaped” People, Not “T-Shaped” | **Autor:** Sarah DaVanzo | **Enlace:** http://bit.ly/2XCtdUu

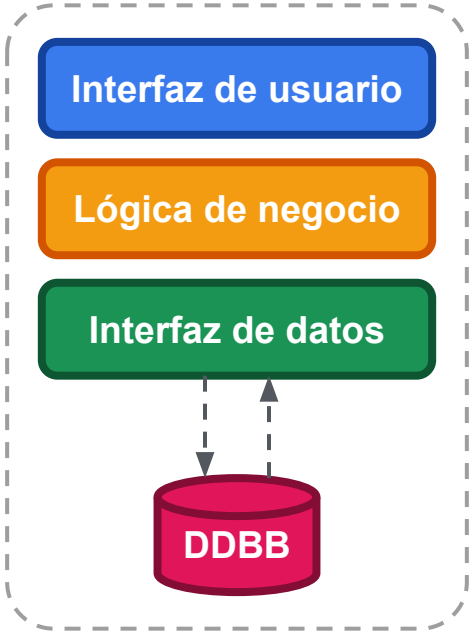
~ 3.2.3 ~

Arquitecturas desacopladas

Arquitecturas desacopladas

En una arquitectura acoplada, pequeños cambios pueden suponer fallos a gran escala. Todos los equipos tienen dependencias entre ellos, además de arrastrar procesos de gestión complejos y burocráticos. Cualquier prueba del sistema supone largos tiempos de espera, y esto deriva en una baja productividad del desarrollador, así como en despliegues con malos resultados.

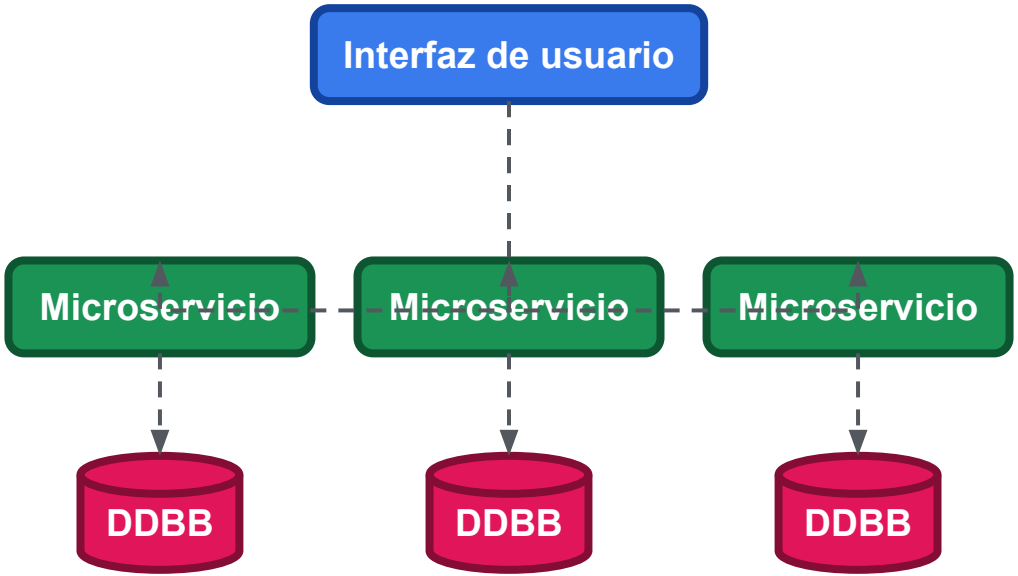
Arquitecturas desacopladas



Arquitecturas desacopladas

Una arquitectura desacoplada permite trabajar y desplegar un servicio de manera independiente; de esta forma, un desarrollador puede actualizar un servicio sin necesidad de conocer cómo funcionan internamente otros servicios de la solución. Esto mejora la productividad del desarrollador.

Arquitecturas desacopladas



~ 3.2.4 ~

Trabajo con equipos pequeños

Two-Pizza Team

La regla *Two-Pizza Team* fue acuñada por Jeff Bezos, CEO de Amazon, para establecer el tamaño ideal de un equipo de trabajo: uno que pueda ser alimentado con dos pizzas. Por lo general, de 5 a 10 miembros.

Two-Pizza Team

Limitar el tamaño de un equipo tiene los siguientes efectos:

- Asegura que el equipo comprende de manera clara y compartida el sistema en el que está trabajando.
- Limita el ritmo de crecimiento del producto en el que se trabaja.
- Posibilita que los equipos sean autónomos.
- Ayuda a los miembros a ganar experiencia de liderazgo en un entorno en el que los fallos no tienen consecuencias catastróficas.

Two-Pizza Team

Site: SlideShare | **Post:** The Future of Enterprise IT - AWS re:Invent 2018 | **Autor:** Sandy Carter | **Enlace:** http://bit.ly/2XZ9Bd2

~ Resumen 3.2 ~

La orientación a producto que ofrece el arquetipo market-oriented parece ser de gran ayuda en la elaboración de equipos.

Contratación de perfiles full stack para evitar que se formen silos.

Usar arquitecturas desacopladas, tanto a nivel organizativo como al construir los productos.

Trabajar siempre en equipos pequeños.

~ 3.3 ~

Cómo integrar Ops en el trabajo diario

Cómo integrar Ops

Nuestro objetivo es crear una estructura *market-oriented*, de manera que pequeños equipos puedan entregar valor al cliente de manera rápida e independiente. Esto puede ser un reto cuando **Ops** está centralizado en una estructura *functionally-oriented*. Se pueden obtener grandes resultados empleando las siguientes estrategias.

~ 3.3.1 ~

Servicios compartidos

Servicios compartidos

Ops puede crear un set de plataformas y servicios centralizados que cualquier equipo de **Dev** puede usar para ser más productivo. Estas son algunas opciones:

- Servicios para la creación de entornos.
- Pipelines de integración y despliegue.
- Herramientas para hacer testing automatizado.
- Dashboards con telemetrías de entornos productivos.

Servicios compartidos

Todas las plataformas y servicios que se dispongan deben estar automatizados, así como poder usarse bajo demanda. Es importante evitar la necesidad de abrir un ticket para que alguien de manera manual realice el trabajo solicitado.

Servicios compartidos

De esta forma, habilitamos que los equipos de producto tengan aquello que necesitan en el momento deseado, además de reducir las comunicaciones y coordinaciones entre equipos.

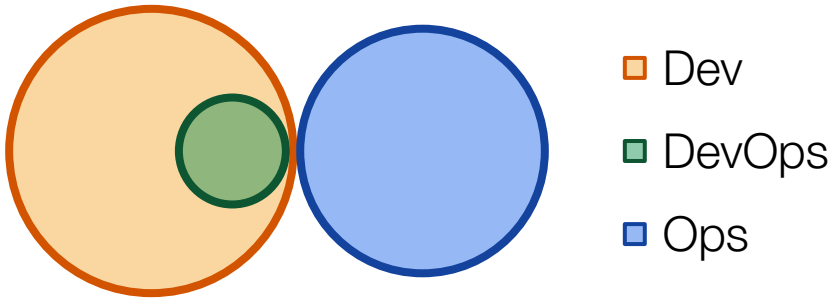
Servicios compartidos

Crear y mantener estas herramientas es similar al desarrollo de un producto real, solo que los clientes son los equipos internos de **Dev**. Si no se atienden las necesidades de este cliente, el resultado será una herramienta pobre que rápidamente será abandonada.

Servicios compartidos

No siempre es necesario construir desde cero: se pueden usar herramientas que tengan buena acogida en la comunidad, en otros equipos e incluso en otras empresas.

Servicios compartidos



~ 3.3.2 ~

Ops dentro de los equipos de producto

Ops en los equipos

Se pueden incorporar ingenieros de **Ops** en los equipos de producto para mejorar su autosuficiencia y reducir la dependencia de un equipo de **Ops** centralizado. Estos equipos de producto pasan a ser responsables no solo de la entrega, sino del soporte del servicio.

Ops en los equipos

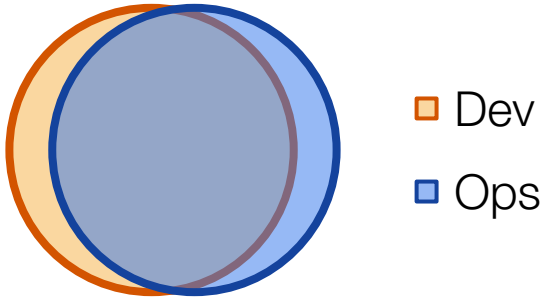
Las prioridades de **Ops** pasan a estar influenciadas por los objetivos del equipo de producto en el que trabajan, ganando así cercanía con sus clientes internos y externos.

Ops en los equipos

Ops participa en todos los rituales del equipo de **Dev**, como los eventos de *planning*, *dailys* y *demos*.

Una vez baja la necesidad de tener capacidades de **Ops** en el equipo, los ingenieros pueden transitar a otros que se encuentren en distinta fase de su ciclo de vida y que sí requieran dicha capacidad.

Ops en los equipos



~ 3.3.3 ~

Ops designado

Ops designado

En aquellos casos en los que no se pueda incorporar ingenieros de **Ops** al equipo de producto, puede designarse un ingeniero desde el equipo centralizado de **Ops** para que atienda las necesidades del equipo de producto.

Ops designado

El ingeniero de **Ops** designado debe atender lo siguiente:

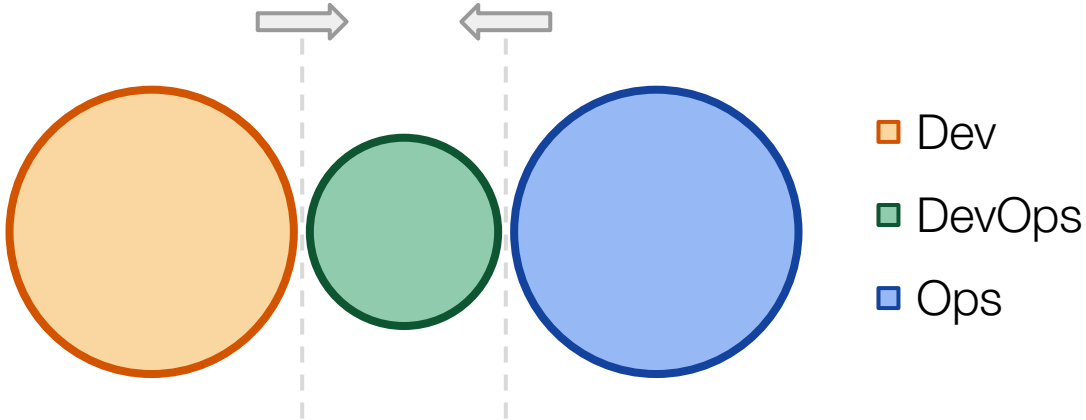
- Cuál es la nueva funcionalidad del producto.
- Si es operable, escalable y observable.
- Cómo se monitoriza y recolectan métricas.
- Si hay cambios en la arquitectura y su justificación.
- Si hay necesidades específicas a nivel de infraestructura.
- Roadmap de lanzamientos.

Ops designado

Como ocurre en el modelo anterior, el ingeniero de **Ops** designado participa en los rituales del equipo de producto; de esta forma puede integrar las necesidades en el roadmap de **Ops** y ejecutar las tareas necesarias.

Tener ingenieros **Ops** designados nos permite atender más equipos de producto, siempre velando por no sobrecargar el modelo.

Ops designado



~ 3.3.4 ~

Ops en los rituales de Dev

Rituales

Cuando los ingenieros de **Ops** están dentro del equipo de producto, o bien cuando se tienen ingenieros **Ops** designados, debemos integrarlos en los rituales del equipo **Dev**.

Esto hace que **Ops** entienda mejor las necesidades del producto y pueda influir con tiempo en el trabajo, antes de que llegue el pase a producción y no haya margen para acometer alguna tarea.

Rituales

- **Daily** → es un evento diario en el cual se expone en qué se trabajó el día anterior, en qué se va a trabajar actual y cuáles son las cosas que impiden que el trabajo salga adelante. La participación de **Ops** en la daily hace que estén al tanto de las actividades de **Dev**, y por tanto puedan planificar y preparar con mayor tiempo.

Rituales

- **Retrospective** → es un evento que tiene lugar al finalizar el sprint. En él se discute sobre qué fue bien, qué puede mejorarse y cómo pueden incorporarse los logros conseguidos en futuras iteraciones o proyectos. **Ops** puede beneficiarse del aprendizaje compartido por otros miembros. Si durante el sprint han tenido lugar despliegues, **Ops** debe compartir los resultados y dar feedback al equipo, para que este entienda cómo impactan las decisiones que se toman.

Rituales

- **Visibilidad del trabajo de Ops** → puesto que **Ops** es parte del flujo de valor del producto, todo el trabajo relacionado con la entrega que sea relevante debe figurar en un tablero kanban compartido. De esta forma identificamos fácilmente qué trabajo se requiere para llevar nuestro código a producción, así como las necesidades de soporte que tiene. También hacemos visible los posibles bloqueos de **Ops** y dónde y cómo podemos darle solución.

~ Resumen 3.3 ~

Ops y **Dev** deben integrarse, directa o indirectamente.

Ops debe conocer en todo momento las necesidades y bloqueos de **Dev** para poder actuar con el debido tiempo.

Ops debe participar activamente en todos los rituales de **Dev**.