UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

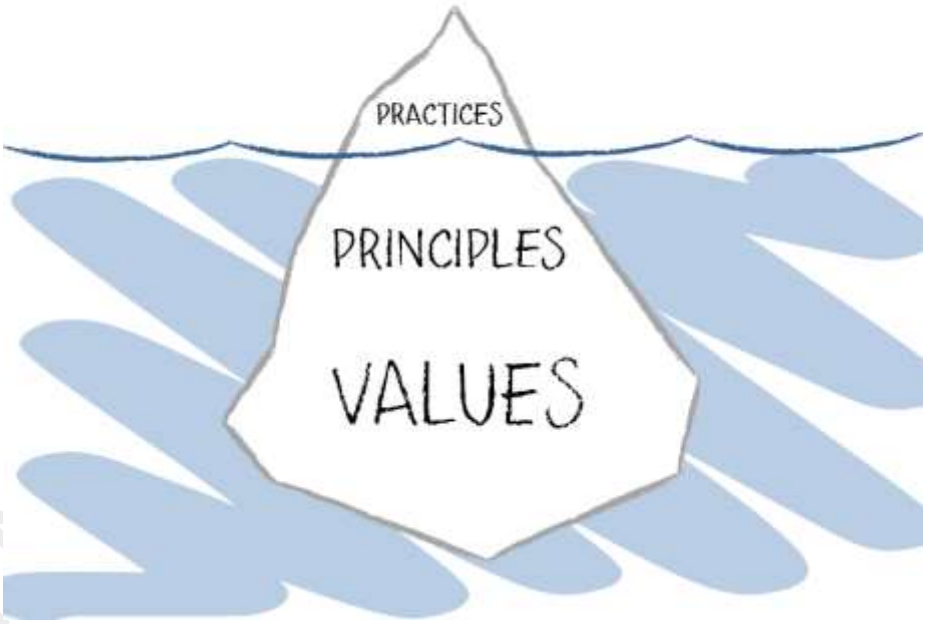
Centro de e-Learning

Curso DevOps, Integración y Agilidad Continúa

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 1 : Unidad 1

Unidad 1: Principios y Valores



Presentación:

En esta unidad se hará una apreciación a los distintos enfoques en los que se sustenta el movimiento de DevOps, se abordará los conceptos fundacionales tanto de la filosofía y principios que de sus bases.

Además se verán conceptos básicos del paradigma de silos funcionales en las organizaciones el cual conlleva ciertos obstáculos al momento de adoptar una cultura DevOps.

Objetivos:

Que los participantes:

- Mencionen e identifiquen los principios de DevOps, Lean y Agile
- Clasifiquen y analicen los componentes de cada propuesta.
- Describan los ciclos involucrados
- Comparen los principales elementos de cada filosofía

Bloques temáticos:

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

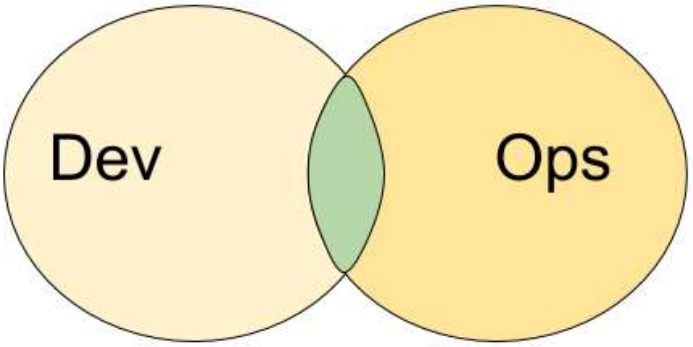
Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

Introducción a DevOps

Filosofía

Comenzaremos con lo que podría darse como una definición, ya que hay muchas y distintas, de lo que es DevOps. Este concepto puede estar enmarcado como un movimiento basado en los principios Lean y en el manifiesto ágil, para abordar la colaboración del área de desarrollo con las áreas de operaciones, y tiene como visión la ***entrega temprana, de valor y calidad a los clientes***, de la combinación del nombre de dichas áreas se creó la nominación DevOps.
Surge como respuesta a los inconvenientes intrínsecos de las estructuras organizacionales basadas en silos funcionales, en este caso puntual las mencionadas áreas de desarrollo y operaciones, que buscan la eficiencia de cada una de ellas pero no así en el sistema completo de la organización. Estas dos áreas tienen objetivos que tienden a poseen fricción en el ciclo completo de entrega: Desarrollo tiene que entregar más software a producción, es decir modificarlo y Operaciones tiene que mantener como están los ambientes.

Entonces, **¿Qué no es DevOps?**. No es una persona en particular, no es un desarrollador haciendo el trabajo de operaciones, tampoco lo es al revés, una persona de operaciones realizando la actividad de un desarrollador, tampoco es un conjunto de herramientas avanzadas ni un rol particular. Es una corriente, un movimiento que se basa en la **colaboración** por parte de las personas que integran, principalmente, las dos áreas fundamentales en TI (Tecnología de la Información) que es **Desarrollo** y **Operaciones**, no obviar que esto también incluye a Seguridad Informática. El movimiento se basa en crear una cultura de colaboración para acelerar los tiempos de entrega de software en manos de cliente/usuario incrementando su calidad. Para ello se basa en una visión y una serie de principios los cuales irán modificando y transformando la organización con el soporte de las herramientas y las prácticas que se ven hoy en día, y algunas de las cuales veremos a lo largo del curso. La filosofía de DevOps lo que intenta promover son comportamientos y valores que posee la organización y las personas que la integran. Sin estos es poco probable que dichas personas adopten las prácticas para las cuales varias herramientas fueron diseñadas.



Por lo tanto, la incorporación de una serie avanzada de herramientas no instalará una cultura DevOps en la organización, pero sí podría verse como una aproximación a que los equipos puedan interactuar de otra manera (introducción de una práctica soportada por una herramienta) pero las herramientas por sí solas no podrán alentar a un cambio cultural en la organización. Dicha interacción es la que facilita el pulido de las asperezas que existen entre Desarrollo y Operaciones, beneficiando a la organización con entregas tempranas y de calidad de software a sus usuarios.

Componentes principales

Consideramos que existen una serie de componentes habilitantes que propician los cambios necesarios para que se puedan adoptar prácticas y cambios de comportamiento.

Comunicación abierta

La cultura de la organización se debe basar en el debate y en la discusión a través de canales claros de comunicación, evitando lo más posible medios, herramientas y procesos que no alienten esto. Por ejemplo, sistema de tickets para implementar, procesos y flujos documentados nunca conversados cara a cara, cadenas de mails, minutas detalladas, etc. Los equipos y las organizaciones vinculadas a DevOps debaten, por ejemplo sobre el tiempo total de entrega del producto (life lead time), la calidad del software, qué construir, medición del impacto de lo que se entrega, cómo hacer esta medición, hacer un uso más eficiente de los recursos, comportamientos en el consumo de la infraestructura, etc.

Alinear los incentivos y la responsabilidad

Los equipos y las diferentes áreas deben estar propulsadas por un propósito, por una visión que permita alinear los objetivos específicos de cada una de ellas. Esa visión es la de construir productos de alta calidad, con un alto valor para el cliente y entregarlos de la manera más rápida posible y con el menor riesgo asociado. Establecer metas de este estilo tiende a diluir la suboptimización de cada área en la organización. Los desarrolladores no serán los únicos responsables por generar el código para que el producto sea bueno, las personas de operaciones no serán castigados por hacer un despliegue a producción que no ocurrió como se esperaba y los testers no serán los únicos responsables de agregar calidad. En lugar de esto, todos tienen la objetivo y responsabilidad de construir el producto lo mejor posible y todos aportan en la creación de un ambiente, un estructura que promueva la colaboración, valiéndose de los éxitos y de los fracasos en conjunto.

You build it, you run it

Es una frase que se utiliza en Netflix para los equipos de producto

Respeto

Los miembros de la organización se deben respetar unos a otros, no importa si están en el mismo equipo o no. No es necesario que una persona le agrade a la otra pero sí que entre ellas puedan trabajar en un ambiente de respeto y apertura hacia distintos puntos de vista. Es necesario cambiar el paradigma tradicional que el respeto y la comunicación va por el camino trazado por el organigrama (en general de abajo hacia arriba) y considerar a cada persona por el rol que cumple, por el valor que aportar, por su desempeño y por su actitudes.

Confianza

Este es un componente fundamental para cualquier tipo de relaciones que quieran alcanzar objetivos comunes. Operaciones debe confiar en que los Desarrolladores harán el mejor código a su alcance, los Desarrolladores deben confiar en que los Testers no levantan incidentes porque están conspirando contra ellos, los Líder de Proyecto deben confiar en que Operaciones está haciendo su trabajo y el feedback de información que les brinda es los suficientemente objetivo.

El comportamiento que se espera es que salir de la cultura de direccionar acusaciones de culpables ante fallas e ir hacia el aprendizaje, mediante sesiones de análisis postmortem, identificando causas raíces y acciones concretas para anclar el aprendizaje identificado en la organización.

Lean

Historia

En esta sección haremos una breve reseña de la historia de Lean que es parte donde se basaron ciertas prácticas de software. Lean comenzó sobre la industria de la manufactura, en la empresa automotriz Toyota. La atención peculiar que tenía esta empresa fue poner su mirada al sistema científico que aplicó Henry Ford a su industria automotriz la cual hasta la segunda guerra funcionó con un eficiente proceso y flujo de producción.
El inconveniente se hizo visible cuando el mercado quería poder elegir entre distintos modelos y/o colores de automóviles, lo cual no era viable para la línea de producción planteada ni para los procesos de la organización.

Kiichiro Toyoda y Taiichi Ohno, junto a las enseñanzas de Edwards Deming[1], de la empresa Toyota observaron el sistema aplicado por H. Ford e hicieron una adaptación y una evolución al contexto que ellos disponían en su empresa. Las problemáticas que observaron, primeramente, es que el sistema de producción de Ford generaba mucho stock de unidades producidas pero no vendidas, es decir, poseían un gran inventario de automóviles que no estaban entregando valor y además la línea de producción como estaba planteada, producía mucho desperdicio en términos de que los ensambles o integraciones de piezas poseían una tasa alta de defectos. Identificados estos aspectos, el sistema de producción que Toyota se propuso y focalizó la mejora continua de la organización en la reducción del desperdicio (Muda, concepto que se verá más adelante), la mejora de los procesos, el adecuado tamaño de la capacidad de la maquinaria, el empoderamiento de las personas que integran la organización (Kaizen[2]), prestando especial atención a la calidad del producto que se entrega al cliente, la entrega justo a tiempo (JIT, just in time) y la

[1] Círculo de Deming, Wikipedia. Disponible desde: https://es.wikipedia.org/wiki/C%C3%ADrculo_de_Deming
[2] Kaizen, Wikipedia. Disponible desde: https://es.wikipedia.org/wiki/Kaizen

automatización (Jidoka) entre otros.

El término “Lean” Manufactura surgió a partir del aporte de los investigadores del MIT (Massachusetts Institute of Technology) James P. Womack y Daniel T. Jones, entre otros autores, al estudiar los sistemas de producción automotrices y en especial el de Toyota. Se suele traducir como magro o libre de grasa.

Principios

El aporte del MIT distinguió cinco principios que el *Sistema de Producción Toyota* [3](TPS, por sus siglas en inglés Toyota Production System) se baso a lo largo de los años en el cual afianzó sus prácticas y técnicas, así mismo la industria del software y de otros rubros han incorporado estos principios para mejorar las sustentabilidad, rentabilidad y el clima en sus organizaciones.

Determinar el valor - **Value**
Se refiere a la necesidad del cliente, lo que le aporta valor. Determinar qué es lo que le dará valor al cliente es el primer principio de Lean, Customer First Thinking (primero pensar en el cliente). Focalizar y orientar la organización para que todo lo que se construye sea para satisfacer las necesidades del cliente.

Identificar el flujo de valor - **Value Stream**
En base al valor detectado que se quiere entregar al cliente, hay que identificar el flujo o los pasos que la organización debe seguir y completar para poder satisfacer la necesidad detectada. Se propone identificar el flujo completo que se desea realizar para entregar el producto. Esto significa que hay que determinar y acordar la secuencia de pasos desde el principio hasta el final, la cual es la entrega del producto al cliente.

Generar el flujo - **Create Flow**
Lo esencial aquí es generar un flujo para que el producto, desde principio a fin, en la cadena de producción se le esté continuamente agregando valor, en otras palabras es hacer

[3] Taiichi Ohno, The Toyota Production System: Beyond Large-Scale Production. 1era edición. Japón,Tokyo: Diamond; 1988

explícito el paso anterior y sentar la base para el principio **pull**. Esto implica que pueden ser necesarias medidas tales como una reestructuración de la compañía o medidas con menor impacto interno como mejorar algún procedimiento o cambiar objetivos en algunas áreas.

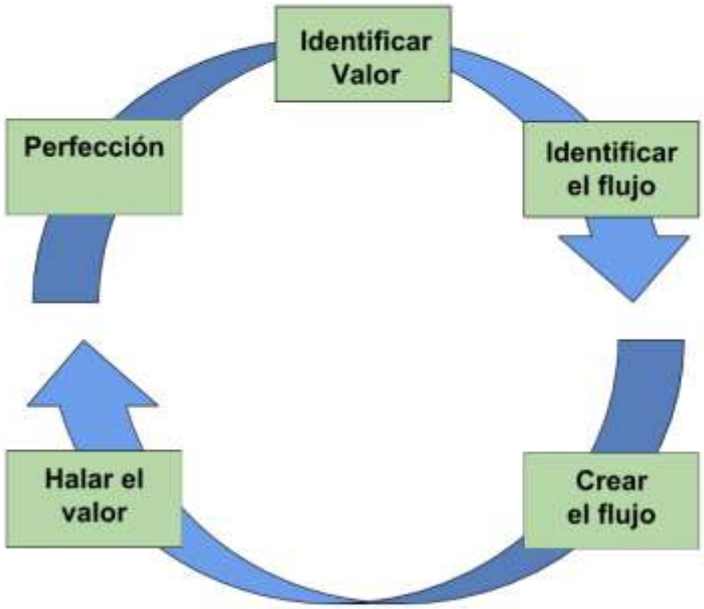
Tirar - **Pull**

El principio pull propone que el sistema de producción comienza cuando hay demanda real de parte de nuestros clientes sin saturar los stocks y capacidades internas de la organización, por lo cual cuando una estación de trabajo termina una pieza sólo se la entregará a la siguiente cuando esta última posea capacidad para poder operar y a su vez, la primer estación no va a tomar/solicitar material nuevo si aún no finalizó con el trabajo en progreso que tiene. Esto beneficia aspectos como la reducción de la cantidad de elementos en los inventarios, la disminución de tiempos muertos entre la ejecución de un proceso y otro, y a identificar los desperdicios que producen los flujos de trabajo que son necesarios para la construcción del producto.

Este principio contrasta a los sistemas tradicionales push que se basan en un control predictivo sobre los inventarios y no sobre las cantidades que se necesitan.

Perfección - **Perfection**

La mejora continua es la núcleo y la visión de Lean. Se mejora continuamente, lo que se definió en un principio se puede mejorar; el proceso, la calidad, la moral de los empleados. Una organización Lean es un sistema que busca la perfección, como un todo.



DevOps se basa en los principios anteriormente descritos, los cuales forman un ciclo que aplicado con compromiso pueden transformar equipos, líneas de productos y la organización. Los principios a su vez no sólo tienen que ser aplicados a nivel de procesos y personal en la organización sino que también se deben aplicar a ciertas prácticas de ingeniería de desarrollo de software para puedan ayudar a lograr el sustento adecuado, generando productos de calidad, reducir el tiempo de entrega, reducir los costos mediante la eliminación de desperdicios y empoderar al personal en su desarrollo profesional.

Por último, DevOps propone un cambio cultural para lograr una manera de desarrollar y desplegar software más efectiva y eficiente, a través de la formalización de prácticas, técnicas, herramientas y objetivos comunes que involucran varias áreas.

Muda (desperdicio)

Uno de los aspectos más conocidos de Lean es la eliminación de desperdicios, categorizandolos de la siguiente manera, que se pueden recordar por el acrónimo D.O.W.N.T.I.M.E:

D. defects (defectos) defectos en el producto

O. overproduction (sobreproducción). Exceso de producción de unidades. Desarrollar más software del que se pidió.

W. wait (espera). Tiempos de espera entre estación de trabajo. Aprobaciones para despliegues, demoras en las decisiones, lejanía de los equipos de producto con los usuarios de la aplicación.

N. not utilizing talent (no aprovechar el talento)- La no utilización del talento de las personas que trabajan en la organización. Personas que pueden aportar en otros aspectos de la organización, por ejemplo comunidades de práctica para esparcir el conocimiento y especializar ciertas técnicas.

T. transportation (traslado)- transporte de mercadería o materia prima de un lado a otro innecesariamente. Tener más ambiente de los necesarios para desplegar software, estrategias de versionado que mueven código de un lado a otro sin necesidad.

I. inventory (inventario)- inventario de productos sin generar valor, se produce y almacena. Se crea software pero no se implementa o se implementa pero no se utiliza.

M. motion (movimiento) - cualquier traslado de maquinaria y/o personas que no agregue valor es considerado desperdicio. Miembros de equipos que necesitan movilizarse para reunirse.

E. Excess processing (exceso de procesamiento) - exceso de proceso en alguna estación de trabajo del proceso, por ejemplo crear múltiples versiones de la misma tarea. Exceso de reportes administrativos, revisiones innecesarias a tareas.

Los principios Lean de manufactura también fueron aplicados a la industria del software, esto es conocido como Lean Software Development.

Lean Software Development

Introducción

La aplicación de los principios de Lean manufactura en organizaciones con líneas de producción fue tan determinante en su sustentabilidad y beneficios que otras industrias como la del software también la adoptaron.

Las organizaciones vinculadas al desarrollo de software adoptaron los principios de Lean manufactura al ciclo de vida de desarrollo con algunas modificaciones sobre el entendimiento de cuándo empieza ese ciclo, dado que no es lo mismo que una línea de producción en serie de una fábrica. Su adaptación incluye los conceptos de desperdicios que vimos anteriormente pero desde la óptica de la construcción, las pruebas y la entrega o implementación del producto, siempre con foco en la mejora continua de la calidad y en la reducción de riesgos y desperdicios permitiendo reducir los tiempos en la entrega de software, principal ventaja de capacidad organizacional para competir en producto digitales.

Un concepto principal que abordaremos en otras unidades es el de **integración continúa**, sobre esta práctica se apoya como base de DevOps y está relacionada con Lean desde la óptica que sea una **sola línea de flujo la que agrega valor**, detener el desarrollo si se encontró un problema grave que desestabilice la aplicación (en las líneas de producción los operarios tenían potestad para frenar la producción ante inconvenientes) y además focalizar desde la organización la manera de reducir los tiempos y riesgos en las entregas.

Principios de Lean Software Development

Los principios[4] que se identificaron son 7 y además reconoce el ciclo que se mencionó anteriormente: Value - Value Stream - Flow - Pull - Perfection el cual permite, a las organizaciones que lo adoptan, aplicarlo en sus productos y áreas o roles. Estos principios están intrínsecos en la cultura DevOps.

1- **Eliminar desperdicio**
Como hemos visto, Lean se basa en la eliminación de desperdicios o Muda. Quitar las actividades que no agregan valor es fundamental para reducir los tiempos en los ciclos y darle un mejor sentido de propósito a las personas que trabajan en la organización ya que estarán haciendo tareas que son utilizadas y necesarias para la organización.

2- **Construir con calidad**
La mirada se debe tener sobre la calidad, mientras más rápido se agregue calidad al software más barato será para la organización la implementación y ejecución del software, obteniendo una mejor reputación con el cliente, ya que la tasa de defectos tiende a disminuir, así como el riesgo operativo de desplegar a producción.
Las pruebas de software deben adentrarse lo más pronto posible en el desarrollo, incluso en etapas de codificación (TDD, es una práctica que orienta el diseño del desarrollo a través de las pruebas, esto se verá más adelante). Al aparecer un defecto, por ejemplo, sobre un build se debe parar lo que se está haciendo y solucionarlo.

3- **Crear conocimiento**
Este principio da una connotación del cambio de pensamiento que ofrece Lean al tradicional esquema predictivo que tiende a establecer el conocimiento o los requisitos de una manera anticipada, previa a la codificación. La idea es dejar de querer predecir el futuro como si dicha predicción fuera precisa (o verdadera) y en lugar de eso basarse en el aprendizaje que el feedback genera a partir de las entregas de software. Este conocimiento es el que permite disminuir la incertidumbre sobre el impacto que pueda tener el desarrollo del producto con los usuarios, por ello se dice que es importante tener un proceso de desarrollo que enfatice el aprendizaje a través del ciclo de desarrollo pero también es necesario mejorar sistemáticamente el proceso de desarrollo. De esta manera se diferencia de la

[4] Poppendieck, M., Poppendieck, T. Lean software development: an agile toolkit for software development managers. 1era edición. Estados Unidos. Addison Wesley; 2003

tendencia tradicional de crear un “estándar” arraigado a procesos documentados de manera rígida que luego interfieren en la mejora no solo de los equipos de desarrollo sino en la organización entera.

4- **Diferir las decisiones**

Dado que los productos de software son, en general, de alta complejidad la toma de decisiones importantes se deben basar en información conocida y no en predicciones. El principio de “diferir las decisiones” se basa en retrasar lo más posible una decisión importante que no se pueda volver atrás para así tener tiempo de conseguir un mayor aprendizaje de manera de tener un menor grado de incertidumbre, decisiones basadas en datos empíricos. Esto no quiere decir que todas las decisiones se tienen que diferir, sino que solo se debiera hacer con las de mayor impacto o las no reversibles, la cual sí sucede se debería revisar para que las decisiones puedan ser reversibles y con poco impacto. Por ejemplo, revertir un cambio desplegado en producción o agregar/eliminar una funcionalidad en el producto.

Aquí difiere nuevamente de la concepción tradicional en la que la planificación supone que dicho plan es el que hay que cumplir sin modificaciones ya que lo considera como un compromiso.

5- **Entregar lo más pronto posible**

Las entregas tempranas de software son beneficiosas por diversos aspectos. Permiten que la organización obtenga réditos económicos más rápido ya que monetiza más pronto el desarrollo que realizó.

Otro aspecto importante en el cual favorece es la posibilidad de liberar porciones de producto que tengan impacto sobre los clientes y medir ese impacto para luego a través del aprendizaje del feedback obtenido se pueda ajustar el producto y satisfacer una necesidad más concreta (buscar el Outcome).

Además, la entrega temprana de software disminuye la ansiedad de los interesados en el producto (stakeholders) de manera que si solicitaron alguna funcionalidad, lo mejor es que esta esté implementada antes de que cambien de parecer.

Por último, las empresas que más frecuentemente despliegan a producción porciones pequeñas de producto son las que poseen menor riesgo asociado en dicha operación de despliegue, ya que es menos riesgoso liberar pequeños fragmentos de software y revertirlo en caso de falla que si fuese porciones grandes. Es uno de los principios de Continuous Delivery, realizar el proceso de despliegue lo más frecuente posible para mejorarlo, con el objetivo de que sea controlable y confiable.

6- **Respetar a las personas**
Las organizaciones basadas en Lean focalizan mucho en este aspecto debido a que entienden que las personas más capacitadas para el desarrollo de software de calidad son los que lo están haciendo y no otros (personas ajenas al equipo). Los roles jerárquicos de la organización deben promover que las personas que se dedican a la producción y al agregado de valor son fundamentales en la organización y se los debe respetar y apoyar en su desarrollo profesional y personal para que puedan hacer mejor su trabajo. Aquí cambia el paradigma de la organización, en donde los roles de toda la organización deben tener una actitud de liderazgo y promover el respeto por el trabajo de todos para que de esta manera se beneficie todo la organización.

7- **Optimizar toda la organización**
Las organizaciones tradicionales se basan en silos funcionales que poseen un funcionamiento de sub-optimización basado en eficiencia de cada área. De esta manera lo que sucede es que un área y sus procesos se optimizan de tal manera que su producción provoca una saturación en el resto generando un aumento de los costos para la organización. Por ello, lo que propone Lean, y en lo que DevOps adoptará mucho de ello, es que al momento de buscar la optimización se piense el todo de una manera sistémica en la cual la optimización de un sector va a modificar el funcionamiento de otros en la organización y eso hay que considerarlo en el análisis de la variable/KPI que se desea optimizar. Este principio se encuentra ligado al ciclo de Lean.

Lean Startup

El método Lean Startup

Uno de los métodos para la creación de productos en que se puede basar una organización que adopte la cultura DevOps es el método de Lean Startup. Esto consiste en una manera de abordar la construcción de productos con un alto nivel de incertidumbre respecto a si ese producto o servicio tiene demanda en el mercado. El desarrollo de la idea busca con los lanzamientos de porciones de producto la **validación de hipótesis** sobre la necesidad del cliente de consumir dicho producto, mediante ciclos de **feedback**. De esta manera se obtiene un aprendizaje del cliente con la utilización del producto y permite una rápida adaptación.
La adaptación permite a las organizaciones ahorrar costos y producir lo que realmente

necesita el cliente, es decir que no es necesario la finalización completa de todas las funcionalidades que se pensaron con antelación. Por otro lado, si se descubre que determinada porción del producto conlleva a mayor satisfacción en el cliente, la organización puede optar por focalizar en escalar su negocio.
Para los casos de nuevos productos este método es fundamental, sobretodo en organizaciones que son emprendimientos o bien células de una organización que poseen una gran **autonomía**, ya que propone que en el primer lanzamiento del producto se piensen (de manera hipoteca) las funcionalidades esenciales que tiendan a satisfacer las necesidades y expectativas del cliente con una primera versión. Este recorte de funcionalidades al inicio del lanzamiento se lo llama **Mínimo Producto Viable** (MVP, por sus siglas en inglés. Minimum Viable Product), y es un concepto fundamental de Lean Startup. La ventaja de abordar de esta manera la creación de nuevos productos es que sólo se desarrollará lo necesario, ahorrando grandes cantidades de dinero y tiempo (disminución del riesgo del negocio), para luego lanzar el producto y aprender del cliente a través de la utilización real. DevOps proveé las prácticas y herramientas necesarias para hacerlo y medirlo, de esta manera la organización pueda adaptar el producto.

La adaptación del producto o el cambio de rumbo en Lean Startup se lo conoce con el concepto de **pívot**. Si al liberar una versión la hipótesis no es válida, se pueden optar por tres opciones. La primera es la del pivote, es decir adaptar el producto y la organización a lo que nuestro objetivo de clientes necesitan y no a lo que la organización sabe o está acostumbrada a producir. La segunda opción es la de insistir con la misma versión del producto pero cambiando el grupo de clientes que consuman el producto. Y por último, desistir. Se decide no enfrentar el costo ni el esfuerzo de continuar con el ciclo de validación de feedback.

La propuesta innovadora de Lean Startup está en la opción del pivote, donde las organizaciones que puedan adaptarse rápidamente al mercado y mejorar su sostenibilidad mediante la adecuación de su producto e internamente en la organización, logran una ventaja por sobre la competencia. Principalmente este es un mejor marco para cuidar las inversiones iniciales sobre nuevas líneas de productos y servicios.

Ciclo Lean Startup[5]

Lean Startup centra la mirada en el cliente y no en el producto, al contrario de como lo

[5] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 1era edición. Estados Unidos: Crown Pub Inc; 2011

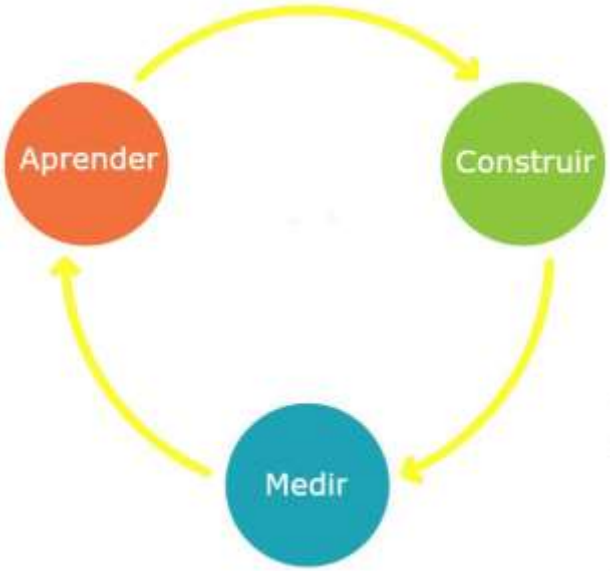
hacen las metodologías tradicionales en las cuales el producto final pasó por todas las etapas del ciclo de vida de desarrollo, sólo permitiendo conocer el producto al finalizar la última etapa.

El ciclo de Lean Startup se basa en tres etapas:

Construir: se desarrolla el producto en base a la hipótesis que se quiere validar. En el caso de una primera versión está será el MVP.

Medir: se establecen una serie de métricas que permitan validar si nuestra hipótesis es verdadera o no.

Aprender: a partir de las métricas que se utilizan para determinar si la hipótesis es válida a partir del feedback que se obtuvo. Esto es posible ya que se obtiene contacto con el cliente, el ciclo se completa con un aprendizaje que puede ser útil para el producto tanto para su visión, su estrategia, o establecer nuevas tácticas.



Este ciclo descrito es iterativo, es decir que para cada hipótesis de versión del producto que queremos lanzar se repite el ciclo de aprendizaje a través de la experimentación con el producto desarrollado.

Agile

Visión

La premisa de las organizaciones TI con orientación a ágiles es la colaboración de los integrantes de los equipos y que estos sean **equipos multidisciplinarios**, cambiar las estructuras hiper jerárquicas por otras más horizontales en donde se pueda tener cierta autonomía y no depender sólo del cumplimiento de objetivos individuales de un área, promover la **auto-organización** y sobretodo el acercamiento e **involucramiento** de los usuarios en el desarrollo del producto junto a las **entregas frecuentes de valor**.
La creación de contextos como los descritos facilita la innovación en las organizaciones entregando productos de alta calidad y de valor a los clientes.
DevOps además de basarse en la filosofía Lean se basa en los principios y valores de la agilidad, junto a sus prácticas y técnicas. Principalmente se basa en los doce principios y los cuatro valores del manifiesto ágil[6].

Ciclos de feedback

Las metodologías ágiles se basan en ciclos cortos (menores a dos meses) en los que permite aprender y adaptar el trabajo realizado a la nueva información que proviene de los usuarios para así mejorar el producto y satisfacer sus necesidades. Mientras más rápido sea el feedback y más veces se lo obtenga en el ciclo de vida del producto mejor, ya que más se aprende sobre el cliente y sobre el proceso de construcción permitiendo mejorarlo. Este aspecto es enteramente tomado por DevOps y Lean Startup para poder obtener un aprendizaje mediante la medición del impacto de la entrega.

[6] Manifiesto Ágil. Disponible desde: http://agilemanifesto.org/iso/es/principles.html

Iteraciones

Los ciclos de feedback, en general, definen las iteraciones de desarrollo o definen las implementaciones. Por ejemplo, un sprint en **Scrum** o un release en **eXtreme Programing**[7] (XP). A medida que los equipos que trabajan en metodologías ágiles alcanzan ciertos grados de madurez tienden a utilizar iteraciones más cortas para tener más competitividad y adaptar sus productos en plazos cortos y así reducir riesgo.

Incremento

En el manifiesto ágil se pueden encontrar ejemplos de la preocupación que existe sobre los despliegues en producción y las demoras en el desarrollo. Entre los doce principios, el de entregar software de manera continua y con valor, o el de entregas tempranas de software funcionando en períodos menores a cuatro semanas son un claro ejemplo de dicha preocupación.
Las metodologías ágiles se basan en el **desarrollo iterativo e incremental** de producto, en un desarrollo orgánico en el que el producto pueda ir creciendo paulatinamente, agregando funcionalidades de manera que lo nuevo no desequilibre lo anterior. Para lograrlo, es fundamental que el desarrollo sea en porciones pequeñas y con foco en la excelencia técnica.
Mediante el incremento de producto y el desarrollo orgánico las metodologías ágiles cambian el paradigma de construcción de software, los ciclos de desarrollo no son como el tradicional en las cuales hay mucha actividad de análisis y diseño antes de la construcción, en la etapa de construcción se codifican todas las piezas y luego se realiza una integración de las mismas dejando las pruebas a lo último. En una iteración el *incremento de producto*[8], es decir la nueva funcionalidad que se desarrolla, tiene que estar completamente probada e integrada al terminar la iteración. Aquí se plantea que el producto crezca como si fuese un organismo vivo, el software se incrementa creciendo como si fuese una planta, y para lograr se basa en las principales prácticas y herramientas de ingeniería de software.

[7] Kent Beck, Extreme Programming Explained: Embrace Change. 2da edición. Estados Unidos, Boston: Addison Wesley; 2004

[8] Para más información apoyarse en la guía Scrum http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Valores

En la agilidad, las prácticas, técnicas y herramientas son soportadas por un conjunto de valores que se desprenden de los doce principios del manifiesto ágil.
Como se mencionó anteriormente, DevOps se basa en los principios Lean y además en los principios y valores del agilismo.

- **Personas e interacciones sobre procesos y herramientas**
- **Software funcionando sobre documentación exhaustiva**
- **Colaboración con el cliente sobre negociación contractual**
- **Adaptación al cambio sobre el seguimiento de un plan**

Para la agilidad no significa que los conceptos resaltados en verde no sean necesarios sino que se les da más importancia a los resaltados en celeste.

Cultura de la organización

Problemas de ambientes

Cuántas veces escuchamos decir a un desarrollador “en mi computadora funciona”. Esto se debe a patrones de comportamiento de la organización en el cual cada persona integrante de un área está aislada de las otras y no poseen una visión sistémica de su trabajo. Es poco probable que las personas vayan todos los días a su trabajo pensando en romper el código de una aplicación, generar mal un despliegue en un ambiente, o crear incidentes solo por fastidiar. Estos rasgos se deben a los comportamientos, castigos, premios que alienta la organización, ya sea consciente o inconscientemente.

En una organización en que se promueven los componentes DevOps que vimos anteriormente, ***Comunicación Abierta, Alinear Incentivos y Responsabilidades, Respeto y la Confianza***, el inconveniente con los ambientes comienza diluirse. Las personas tienden a colaborar para que todos puedan crear productos de calidad para toda la organización y no para cumplir con la única responsabilidad del área funcional (esto en Lean se lo conoce como suboptimización) o en enfocarse en la acción que realizó tal persona y fue juzgada como mala. Además, con la colaboración de diversos roles las organizaciones basadas en DevOps poseen una serie de prácticas y herramientas que permiten resolver estos inconvenientes, en donde el trabajo de las personas de operaciones ayudará a la creación de ambientes para facilitar el desarrollo y las pruebas de software, y viceversa. Las personas del área de Operaciones/Infraestructura, tales como los roles de administradores de sistemas, de redes, de base de datos, personal de seguridad informática, puedan sugerir ciertos cambios al desarrollo para mejorar la calidad, desempeño y facilitar la obtención de métricas para observar el comportamiento del sistema.

Los inconvenientes de ambientes más comunes son en las diferencias de versiones. Por ejemplo, diferencia en la versión de algún aprovisionamiento de software, en el sistema operativo, en la configuración de la aplicación. O bien, diferencias de ambientes por no tener un estándar en su creación, es decir que ese estándar no existe para distribuir el mismo ambiente de desarrollo para todo un equipo.

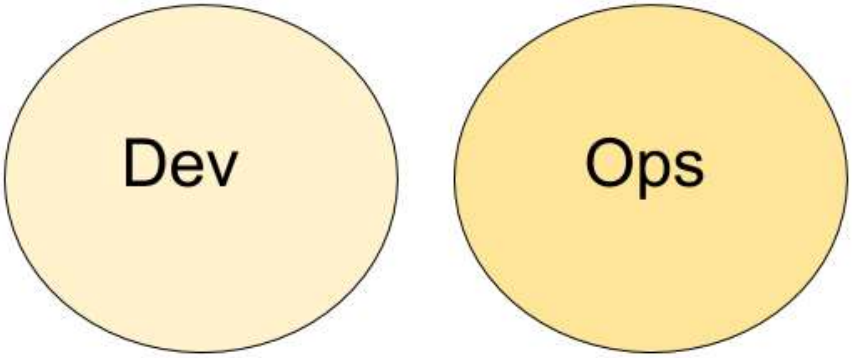
Antipatrones de equipos y organizacionales

Tal como hemos visto a lo largo de esta primer unidad, las organizaciones tienen estructuras que al momento de realizar un acercamiento a DevOps surgen como impedimentos, obstaculizando la adopción de un entorno que favorezca aspectos tales como las entregas de calidad, la colaboración, la satisfacción del cliente, mejor clima laboral, menor riesgo operacional, etc.

Las estructuras organizacionales son estructuras sociales, las cuales tienen diferentes demarcaciones. Estas demarcaciones se presentan, en general, en la conformación de diferentes áreas que tendrán primariamente la asignación de cierta autoridad, coordinación de personas (con su debido reporte), objetivos, incentivos, recompensas y castigos comunes a la organización pero la mayoría de las veces tiene más peso el propio del área. Las divisiones de cada área, por ejemplo Desarrollo, Operaciones, Testing, Project Management, etc. muchas veces no favorecen la colaboración entre las mismas. Aquí haremos una breve reseña de algunos patrones de división de áreas y/o equipos que son propensos a limitar a una organización a la finalidad que tiene DevOps.

Desarrollo y Operaciones separados como silos funcionales

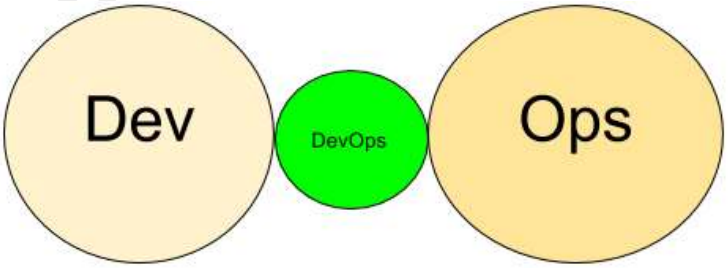
Esta división es la más común que se encuentra en las organizaciones de hoy en día. Hay un área de Desarrollo que realiza el núcleo de funcionalidades más el agregado de calidad con una subárea de Pruebas y por otro lado un área de Operaciones que se encarga de llevar el desarrollo a manos del cliente. El inconveniente radica en que las áreas no colaboran, sino que una vez que el Desarrollo está listo se lo pasan a manos de Operaciones y si algo malo ocurrió inesperadamente esta última vuelve a pasarle la atención del problema, en el mejor de los casos, a Desarrollo. Ambas áreas se comportan como cliente-proveedor dependiendo del momento en que se encuentra el ciclo del producto.



La colaboración entre ambos es fundamental para que los integrantes obtengan conocimiento de aspectos mutuos que son primordiales para el desarrollo y la entrega al cliente. Ejemplos de este probable desconocimiento pueden ser: métricas que toma el equipo de operaciones (tales como usabilidad, tiempo de despliegue, desempeño, etc), configuración de ambientes que no son accesibles por equipo de Desarrollo, configuraciones que podrían aumentar el desempeño pero que no son implantadas.

Equipo DevOps aislado

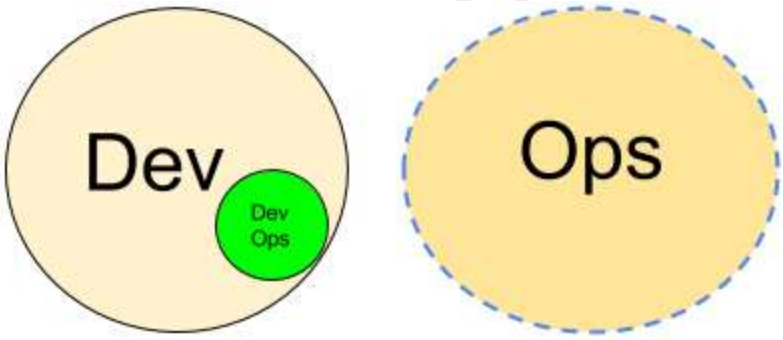
Este patrón de división en líneas generales es adoptado cuando la decisión de adoptar un equipo DevOps proviene de mandos altos en la estructura de la organización. La problemática que surge con esta disposición es que el valor que genera la cédula DevOps no alimenta de conocimiento al resto de la organización, sigue siendo un silo funcional, por lo tanto puede provocar un mayor distanciamiento en las interacciones entre las áreas de Desarrollo y Operaciones.



Esto sólo es recomendado para dar una iniciativa temporal en la organización. Por temporal se entiende entre un plazo de 12 y 18 meses. Sin embargo, es bueno resaltar que para dar una iniciativa así es mejor que la interacción resultante de la célula DevOps actúe como proxy entre las otras dos, de manera tal estas puedan incorporar su aprendizaje.

Desarrollo no necesita de Operaciones

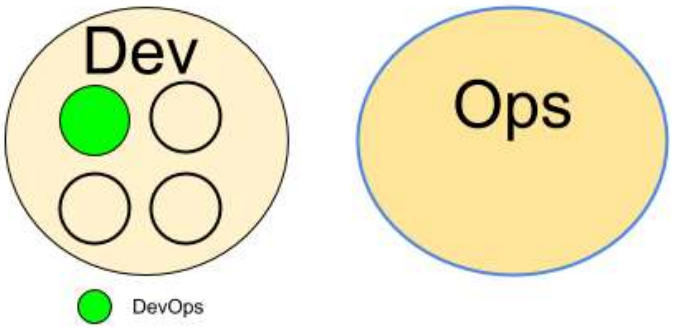
Esta formación probablemente sea a causa de falta de madurez del área de desarrollo o por decisiones del manager de dicha área. La principal problemática es que el área de Desarrollo está subestimando las habilidades que tiene Operaciones de agregar valor al producto (por ejemplo, gestión de licencias y de recursos de infraestructura, guardias, soporte, etc), y además no se tiene una mirada sistémica sobre la organización en donde ignorar la importancia de un área, trae consecuencias directas sobre la calidad de lo que se elabora.
Otra perspectiva de la implicancia de querer esta división es que el área de Desarrollo debería tener las suficientes habilidades poder liberar software de calidad y además actuar ante contingencias, sin frenar el ritmo de desarrollo.



DevOps como un equipo de herramientas

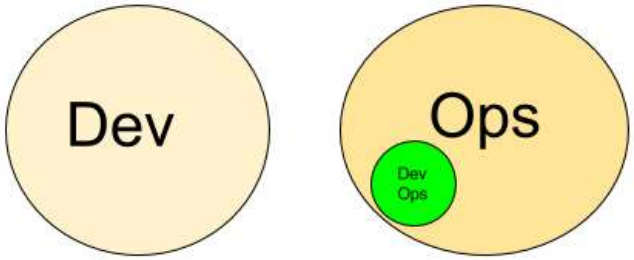
Un patrón que se encuentra comúnmente en las organizaciones es adoptar un equipo especializado en herramientas en las cuales facilitan los procesos de despliegue y desarrollo (pipeline), medición, orquestación y configuración, pero dejando de lado el foco en que se basa el mismo nombre de DevOps, el cual es la colaboración de las áreas de Desarrollo y Operaciones. Además, con este patrón se está directamente excluyendo al

área de Operaciones con lo cual no hay un flujo continuo en que el producto atraviese ambas áreas hasta llegar al cliente sino que solo se soporta en herramientas, lo cual como ya hemos visto por sí solas no posibilitan el cambio necesario en la organización para reducir el tiempo de ciclo de entrega del software.



Re bautizar el rol SysAdmin

Este patrón se presenta bastante en organizaciones con poca madurez y conocimiento en agilidad. Surge, probablemente, de decisiones que son tomadas por mandos superiores, generalmente, con poco conocimiento sobre el tema e incorporan personal especializado en las mejores prácticas de ingeniería en desarrollo de software al área de Operaciones pero con el título de “DevOps”. Este patrón es similar al de “Desarrollo no necesita de Operaciones”. Sólo se optimiza el área de Operaciones sin ningún tipo de interacción con el área de Desarrollo.



Bibliografía utilizada y sugerida

- Jez Humble and David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. 1era edición. Estados Unidos. Addison Wesley; 2010
- Kent Beck. Extreme Programming Explained: Embrace Change. 2da edición. Estados Unidos, Boston: Addison Wesley; 2004
- Mandi Walls. Building a DevOps Culture. 1era edición. Estados Unidos: O'Reilly; 2013
- Matthew Skelton. Pattern of DevOps Culture. InfoQ eMag; 2015; Disponible desde: https://www.infoq.com/minibooks/emag-patterns-devops-culture
- Poppendieck, M., Poppendieck, T. Lean software development: an agile toolkit for software development managers. 1era edición. Estados Unidos. Addison Wesley; 2003
- Ries, Eric. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. 1era edición. Estados Unidos: Crown Pub Inc; 2011
- Taiichi Ohno, The Toyota Production System: Beyond Large-Scale Production. 1era edición. Japón,Tokyo: Diamond; 1988

Lo que vimos:

Una reseña teórica sobre la línea de pensamiento, principios y valores en los cuales se sustenta el movimiento DevOps. Comprender que la fundación del movimiento se basó en conceptos sobre Lean, Agilidad y que luego en base a esta visión y propósito se incorporaron las herramientas. Se pudo observar que el paradigma de silos funcionales la gran mayoría de las veces provoca inconvenientes en los objetivos globales de una organización.

Lo que viene:

En la siguiente unidad veremos las distintas necesidades que se consideran requeridas para iniciar el camino a DevOps. Para ello, se verán ciertos conceptos y técnicas que favorecen y se requieren para la práctica de Integración Continúa.