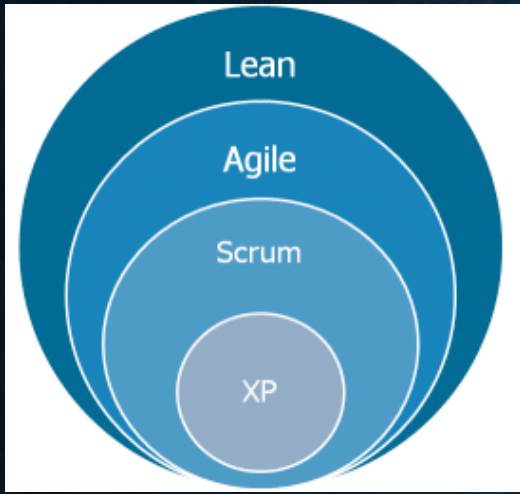


# GESTION DE PROYECTOS

**Agilidad**

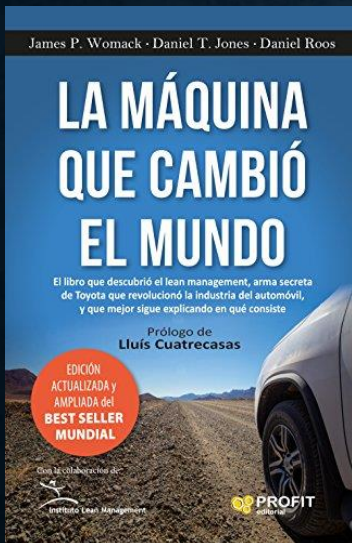
Lic. Virginia Polcan

# PENSAMIENTO LEAN (SIN GRASA)



Lean es un marco filosófico que antecede y fundamenta el pensamiento ágil y sus marcos de trabajo

- ✓ Lean: “sin grasa”, sin desperdicios
  - ✓ LEAN: Toyota Lean Manufacturing basado en :
    - ✓ Eficiencia operativa con la mejora continua y la eliminación del desperdicio.
- Toyota Production System
- TPS da lugar a LSD (Lean Software Development)
- ✓ Agile adapta la filosofía lean para el desarrollo de sw para manejar la incertidumbre



# LEAN SOFTWARE DEVELOPMENT

- 1948/1975 origen TPS (Toyota Production System) Surge el pensamiento Lean Manufacturing para optimizar fabricación de automoviles)
- 2001 Manifiesto ágil para el desarrollo de sw
- 2003 Oficialmente se formaliza el desarrollo de sw Lean con el libro “LSD” de Mary y Tom Poppendieck, adaptando los principios de TPS al SW
- 2011 Eric Ries publica “The Lean Startup” integrando lean al desarrollo ágil de negocios

TRAS LA GUERRA FRÍA

TRAS LA CRISIS SANITARIA COVID 19



1980

Un mundo complejo  
y ambiguo



2020

Un mundo  
caótico

**VUCA**

Volátil  
U-Incierto  
Complejo  
Ambiguo

**BANI**

B-frágil  
Ansiedad  
No lineal  
Incomprensible

# REVOLUCION EN LOS MODELOS DE NEGOCIOS

Nuevos paradigmas de comportamiento  
de los clientes y de los mercados

- Comercio colaborativo (Amazon) 1994
- Streaming de video (Netflix) 1997
- Streaming de musica (Spotify) 2006
- Mercados online de hospedaje (Airbnb) 2008
- Aplicaciones multiplataforma de mensajes instantaneos (WhatsApp) 2009
- Servicios digitales de transporte (Uber) 2009

**1919**  
**107 años**



**1,360,000 Rooms**  
**144 países**

**2008**  
**18 años**



**8,000,000 Rooms**  
**220 países**

# MANIFIESTO PARA EL DESARROLLO ÁGIL DE SOFTWARE

- ✓ Febrero de 2001
- ✓ 17 firmantes, lideres en desarrollo de software agil
- ✓ Firman el llamado:

**Manifiesto de  
desarrollo de software ágil**

- <https://agilemanifesto.org/iso/es/manifesto.html>



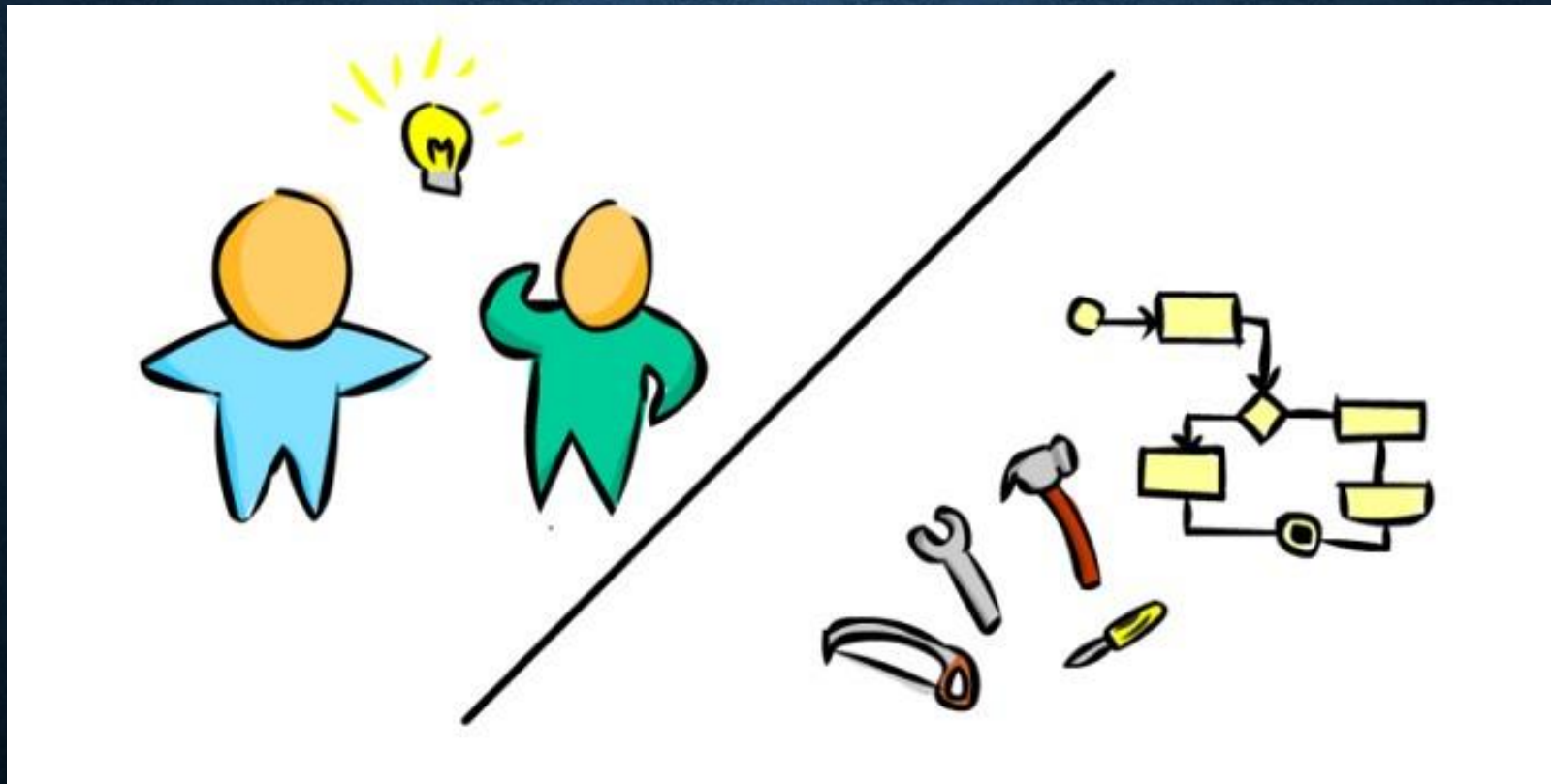
## **4 VALORES**

**Se priorizan:**

- 1. Individuos e interacciones** sobre procesos y herramientas
- 2. Software funcionando** sobre documentación extensiva
- 3. Colaboración con el cliente** sobre negociación contractual
- 4. Respuesta ante el cambio** sobre seguir un plan

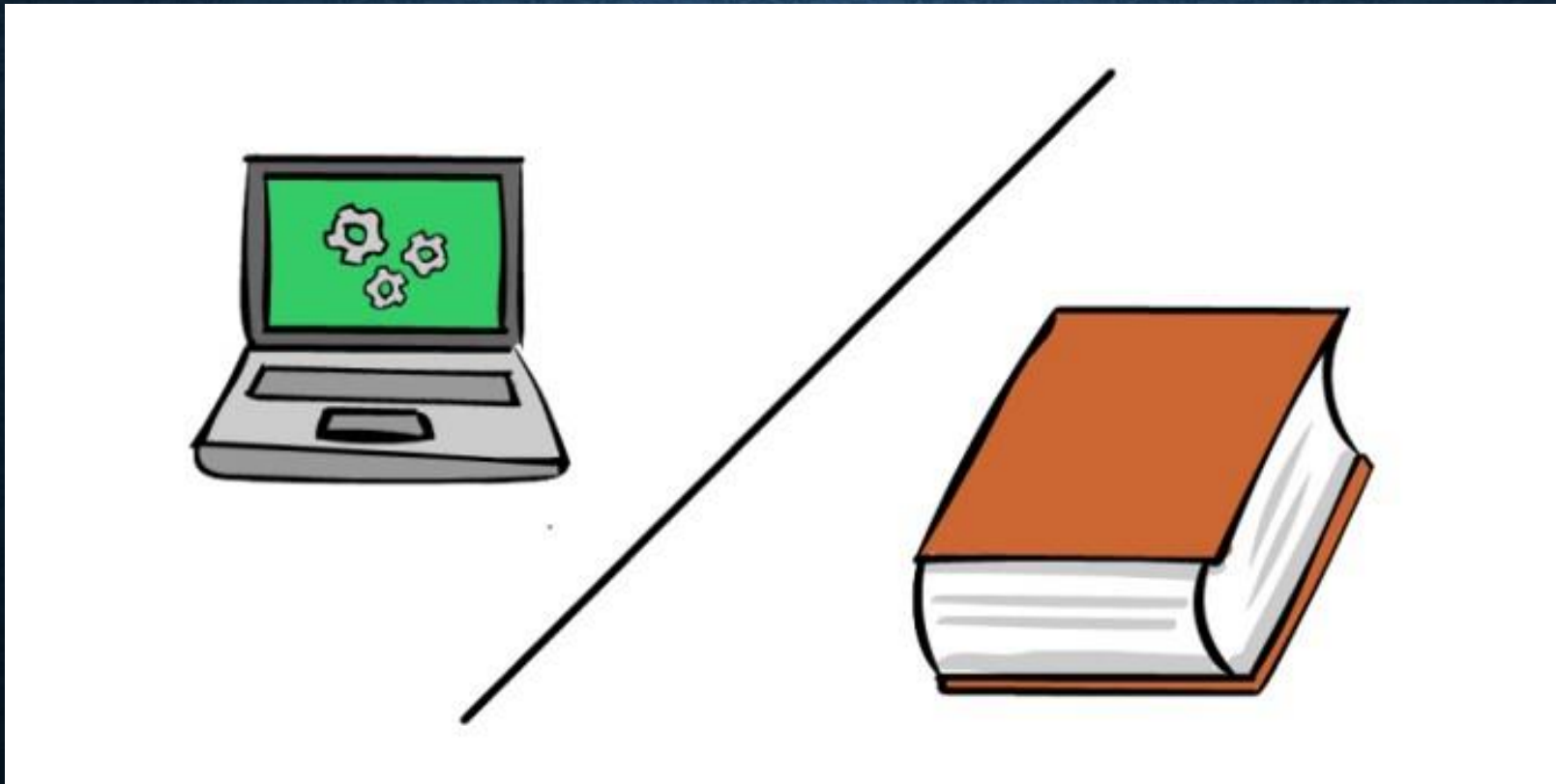
# VALORES

## 1. Individuos e interacciones sobre procesos y herramientas



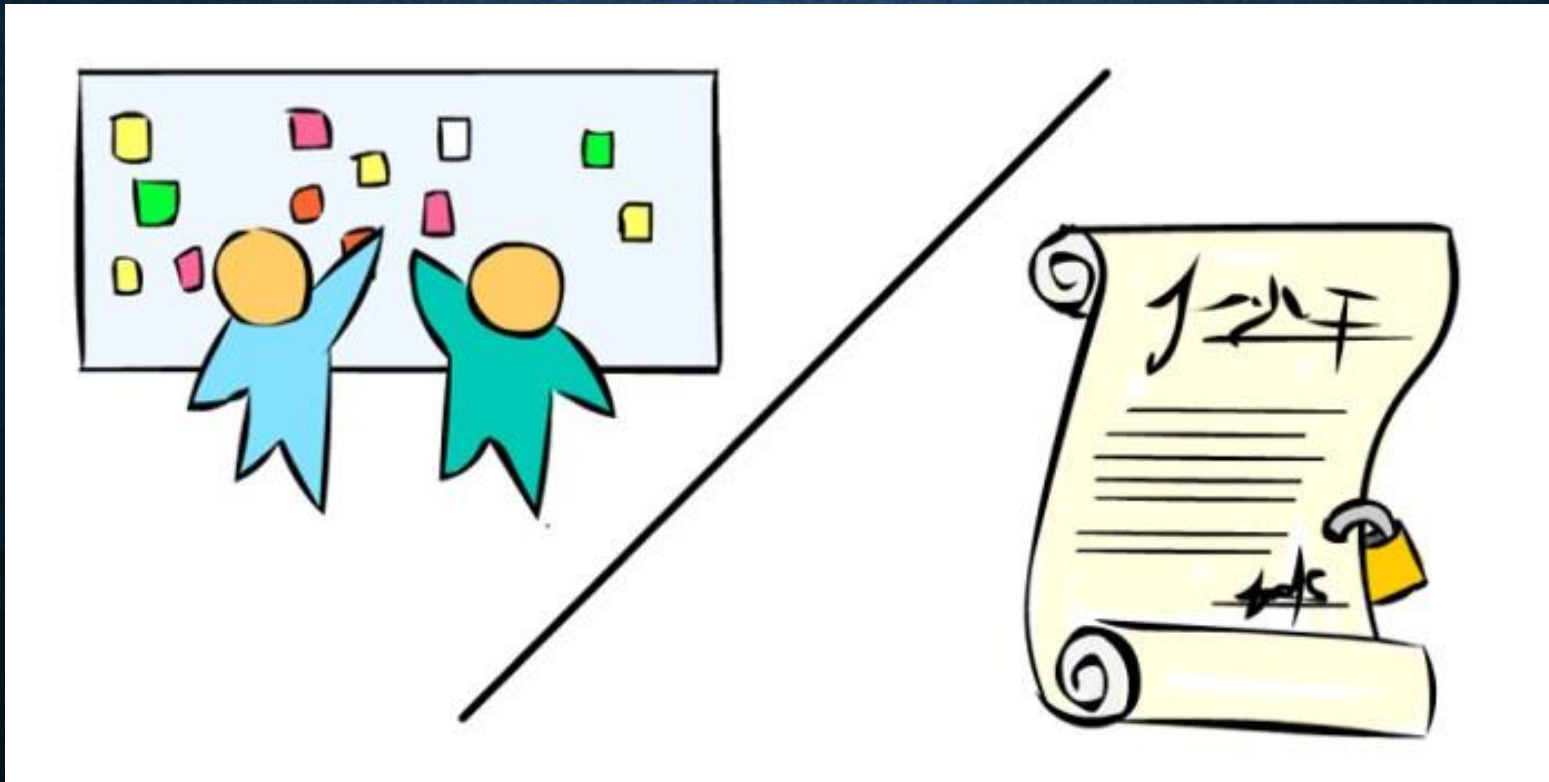
# VALORES

## 2. Software funcionando sobre documentación extensiva



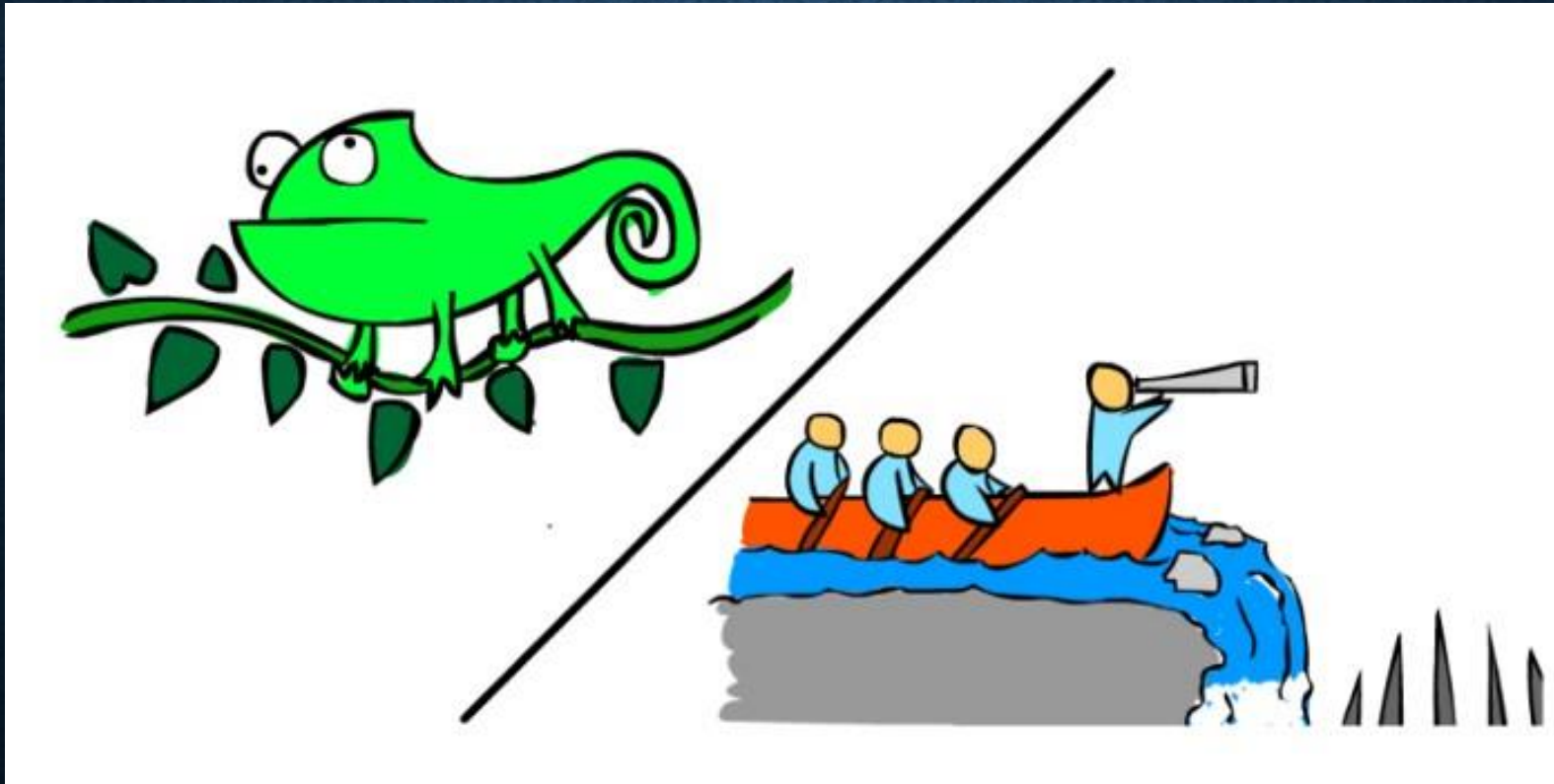
# VALORES

## 3. Colaboración con el cliente sobre negociación contractual



# VALORES

## 4. Respuesta ante el cambio sobre seguir un plan



## 12 PRINCIPIOS

- Los principios son las “recetas” que nos permiten llevar a la práctica los valores.

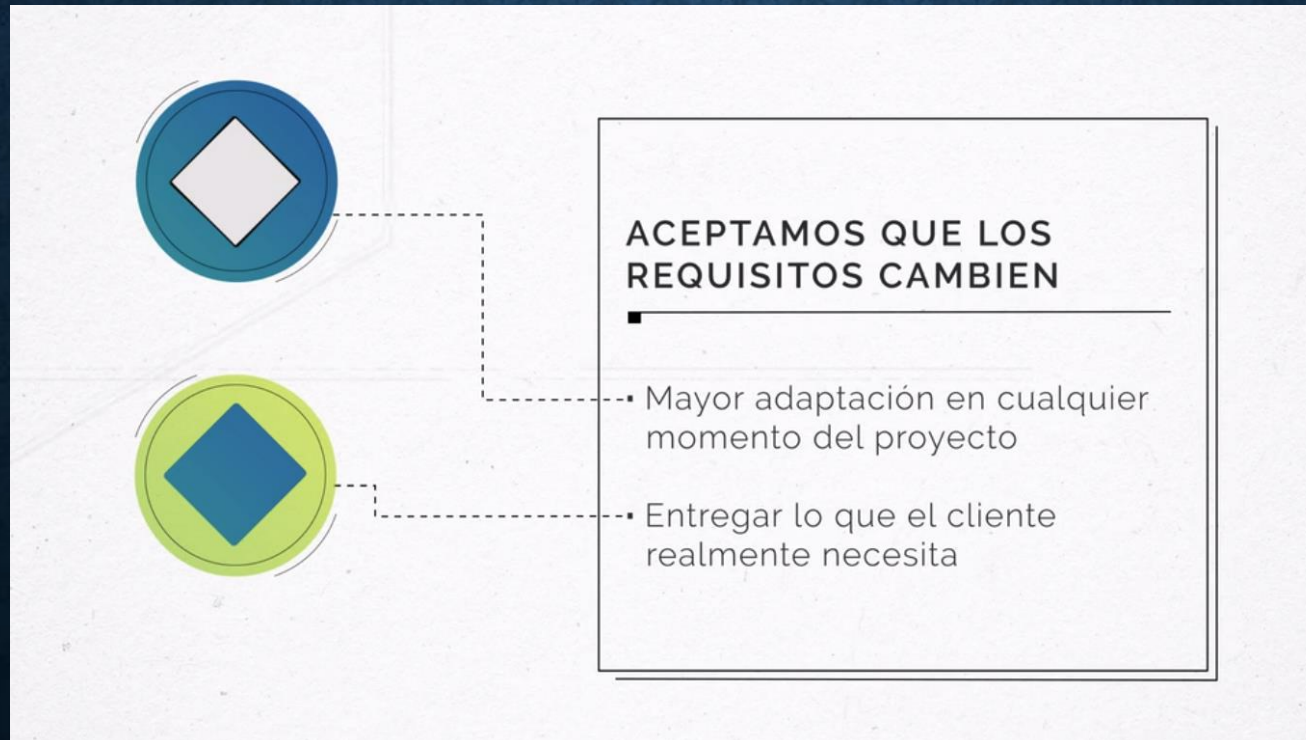
# 1ER PRINCIPIO

Nuestra mayor prioridad es **satisfacer** al cliente mediante la entrega **temprana y continua** de software con **valor**.



## 2DO PRINCIPIO

Aceptamos que los requisitos **cambien**, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar **ventaja competitiva** al cliente.



# 3ER PRINCIPIO

**Entregamos software funcional frecuentemente**, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.



## ENTREGAR SOFTWARE FUNCIONAL FRECUENTEMENTE

- Feedback continuo
- ¿Vamos bien?
- ¿Qué necesitas?

# 4TO PRINCIPIO

Los responsables de negocio y los desarrolladores **trabajamos juntos de forma cotidiana** durante todo el proyecto.



ES NECESARIO QUE EL ÁREA  
DE NEGOCIO E IT TRABAJEN  
CONJUNTAMENTE

- Mayor colaboración
- Mayor confianza entre miembros del equipo

# 5TO PRINCIPIO

Los proyectos se desarrollan en torno a **individuos motivados**. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.



## LOS MEJORES PROYECTOS SE DESARROLLAN CON INDIVIDUOS MOTIVADOS

- Mayor motivación
- Satisfacer las necesidades de las personas

# 6TO PRINCIPIO

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la **conversación cara a cara**.



CARA A CARA ES LA MEJOR  
MANERA DE COMUNICAR

IMPORTANTE MANTENER LA  
TEMPERATURA DE LA  
COMUNICACIÓN LO MÁS ALTA

- Evitar malos entendidos
- Llegar antes a un acuerdo
- Importancia de la comunicación no verbal

# 7MO PRINCIPIO

El software **funcionando** es la principal **medida de progreso**.



# 8VO PRINCIPIO

Los procesos Ágiles promueven el desarrollo **sostenible**. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.



LOS ENFOQUES ÁGILES  
PROMUEVEN EL DESARROLLO  
SOSTENIBLE

- Sostenible --> Previsible
- No sobrecargarnos de trabajo

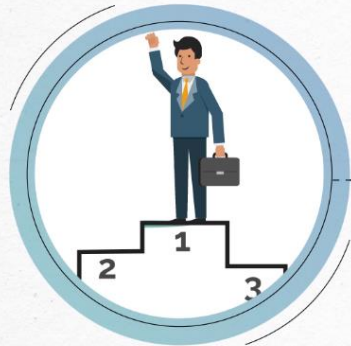
# PRINCIPIO NRO 9

**La simplicidad**, o el arte de maximizar la cantidad de trabajo no realizado, es **esencial**.



# PRINCIPIO NRO 10

La atención continua a la **excelencia técnica** y al buen diseño mejora la Agilidad.



ATENCIÓN CONTINUA  
+ EXCELENCIA TÉCNICA  
+ BUEN DISEÑO  
= MEJORA DE LA AGILIDAD

- Calidad desde el principio del proyecto
- Simplificación de mantenimientos
- Vivir más tranquilos y felices

# PRINCIPIO NRO 11

Las mejores arquitecturas, requisitos y diseños **emergen** de equipos **auto-organizados**.

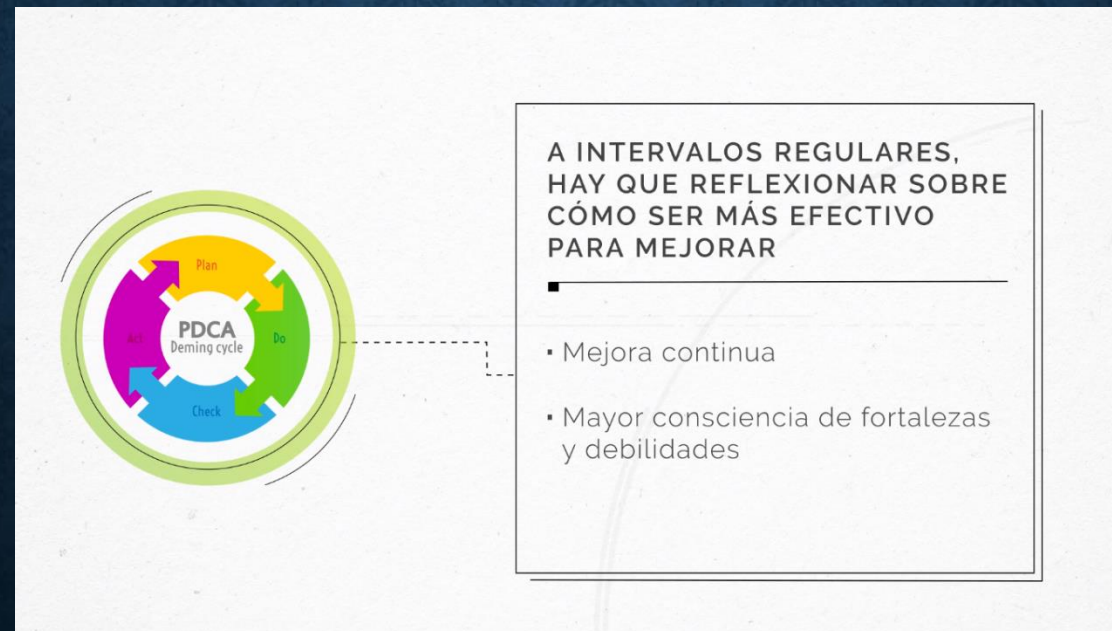


LAS MEJORES  
ARQUITECTURAS, REQUISITOS  
Y DISEÑOS EMERGEN DE  
EQUIPOS AUTORGANIZADOS

- Mayor responsabilidad + Mayor autonomía + Mayor conciencia = Aumento de la motivación
- Decidir cómo hacer las cosas, por los que mejor saben hacerlas

# PRINCIPIO NRO 12

**A intervalos regulares** el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.



# LEAN SOFTWARE DEVELOPMENT

- 1948/1975 origen TPS (Toyota Production System) Surge el pensamiento Lean Manufacturing para optimizar fabricación de automoviles)
- 2001 Manifiesto ágil para el desarrollo de sw
- 2003 Oficialmente se formaliza el desarrollo de sw Lean con el libro “LSD” de Mary y Tom Poppendieck, adaptando los principios de TPS al SW
- 2011 Eric Ries publica “The Lean Startup” integrando lean al desarrollo ágil de negocios

# LSD – DESARROLLO DE SOFTWARE LEAN



*Ilustración 1 – Principios Lean Software Development (LSD)*

# ELIMINAR DESPERDICIOS

- Por desperdicio o basura consideramos todo aquello que no aporta valor al cliente.
- Funcionalidades no necesarias
- Retrasos en el proceso de desarrollo de software
- Mala toma de requisitos
- Problemas con la comunicación interna

# APRENDIZAJE CONTINUO

- Es de vital importancia que todos los miembros del equipo de desarrollo trabajen con una mentalidad de aprendizaje continuo.
- El hecho de que un desarrollador trabaje con una tecnología o lenguaje concreto no quiere decir que no pueda aprender de otros compañeros o proyectos.

# TOMAR DECISIONES LO MAS TARDE POSIBLE

- En un modo tradicional o cascada, el proyecto parte de unos requisitos iniciales que condicionan todo el ciclo de desarrollo de software (alcance congelado) y cualquier cambio plantea replanificación y adición de trabajo al proyecto.
- En una filosofía Lean, Agile generalmente, los requisitos suelen ser sustituidos por *historias de usuario* que están más cerca de la necesidad real del cliente.
- Por este motivo, podemos esperar a construir el software hasta que la *user story* esté definida claramente y sin ambigüedades.

# ENTREGAR LO ANTES POSIBLE

- En un modelo Lean, las entregas de software son más frecuentes incluyendo *features* (*funcionalidades o características*) alineadas con las *user stories*.
- Cada entrega incluirá funcionalidades que necesitan los usuarios lo antes posible basadas en prioridades de desarrollo.

# POTENCIAR EL EQUIPO

- Facilitar que los desarrolladores participen en la toma de decisiones de tiempos asociados a tareas, priorización de las mismas. Esto hace que los miembros del equipo se sientan parte importante en él.
- Además, los propios desarrolladores saben, de primera mano, qué tareas cuestan más, cuales menos y qué implicaciones tienen en el ciclo de vida del proyecto.
- Técnicas como el *Planning Poker* en las que se asignan pesos, esfuerzos o complejidades a las tareas a realizar caen dentro de este principio.

# CREAR LA INTEGRIDAD

- Contar con un buen sistema de integración continua que incluya pruebas automatizadas, pruebas de usabilidad, son críticas para que un software sea fácil de mantener, de mejorar y de reutilizar.
- Con esto evitaremos añadir "**basura**" a dicho software e intentar aprovechar lo aprendido de proyectos anteriores.

# VISUALIZAR TODO EL CONJUNTO

- Visualizar el contexto
- Pensar en grande, actuar en pequeño
- Equivocarse rápido, equivocarse pronto es equivocarse barato.
- Equivocarse rápido y aprender con rapidez.

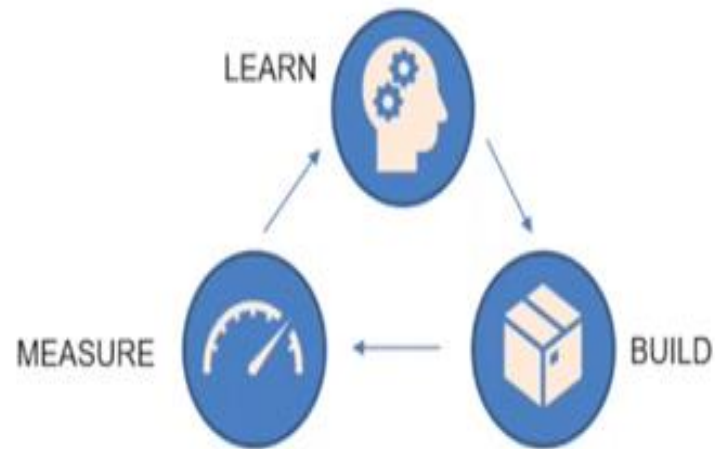
# LEAN SOFTWARE DEVELOPMENT

- 1948/1975 origen TPS (Toyota Production System) Surge el pensamiento Lean Manufacturing para optimizar fabricación de automoviles)
- 2001 Manifiesto ágil para el desarrollo de sw
- 2003 Oficialmente se formaliza el desarrollo de sw Lean con el libro “LSD” de Mary y Tom Poppendieck, adaptando los principios de TPS al SW
- 2011 Eric Ries publica “The Lean Startup” integrando lean al desarrollo ágil de negocios

# Que es Lean Startup

La metodología permite acortar los ciclos de desarrollo de productos, adoptando una combinación de:

- experimentación impulsada por hipótesis para medir el progreso,
- lanzamientos de productos iterativos para obtener feedback de clientes y
- aprendizaje validado para medir cuánto se ha aprendido.

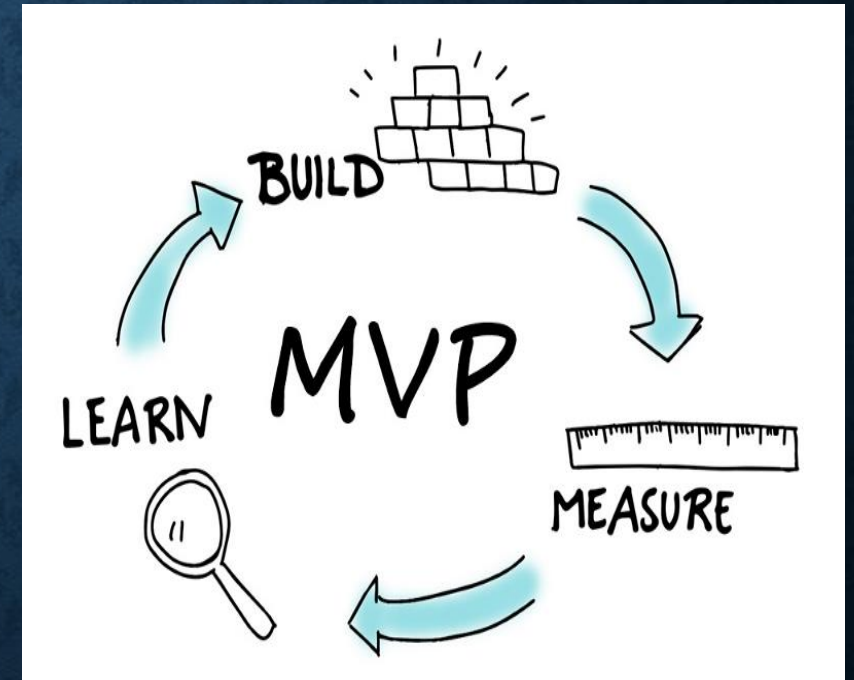


# LEAN STARTUP (ERIC RIES)

1. Supuestos de fe (hipotesis)
2. Experimento (MVP)
3. Aprendizaje validado
4. Retroalimentacion (crear-medir-aprender)
5. Periodicamente:
  1. Pivotar (cambiar de estrategia)
  2. Perseverar (mantener el rumbo)

# CICLO DEL MVP

- **CONSTRUIR:** construir un producto minimo viable con las funciones esenciales para probar el mercado
- **MEDIR:** Establecer metricas para concluir el resultado del experimento (ej: cantidad de registros, numero de visitas, cantidad de pre-ventas, numero de descargas, etc)
- **APRENDER:** Se presenta el MVP al publico y las lecciones aprendidas. Con base a ese aprendizaje, volver al paso 1, modificando o pivotando el MVP y volver a empezar



# CARACTERISTICAS DEL MVP

- Tiene valor en cuanto que puede ser utilizado
- Version mínima
- Tiene funcionalidades minimas o imprescindibles que ya plantean una solucion
- Permite poner a prueba el producto y captar los primeros usuarios
- Da posibilidad de obtener feedback
- Provee informacion sobre los clientes: que funcionalidades o soluciones son las que mas valora la buyer persona
- No es un producto incompleto o de baja calidad sino que es viable, es decir, resuelve un problema real para obtener aprendizaje validado

# PRINCIPIOS LEAN

## ➤ **Entrega de valor continuo**

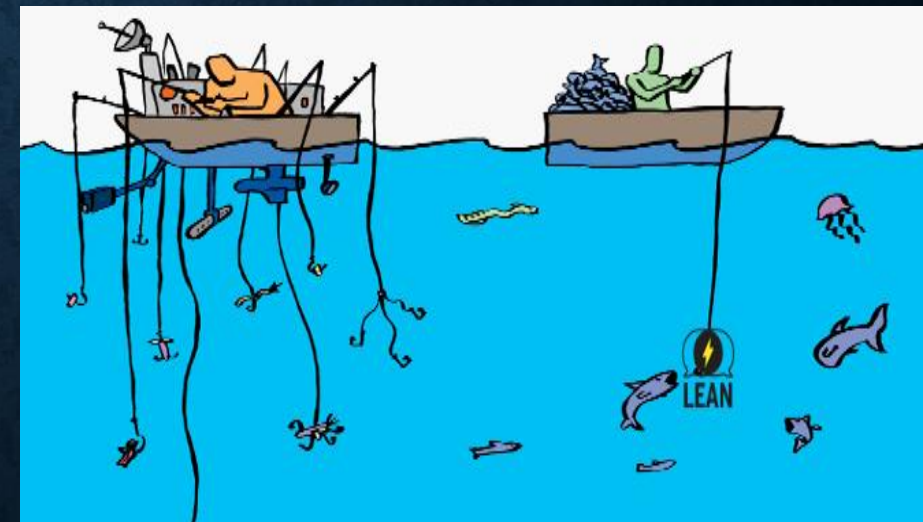
- *Iterativo: se va construyendo en pequeños ciclos de vida y se recibe feedback del cliente en cada iteración, lo que permite refinar el producto tras el feedback*
- *Incremental: añadir valor en cada entrega asegurando que cada entrega sea subconjunto funcional del total*

## ➤ **Minimizar el desperdicio**

- *Rework, evitar rehacer tareas*
- *Reducir tareas que no agregan valor*

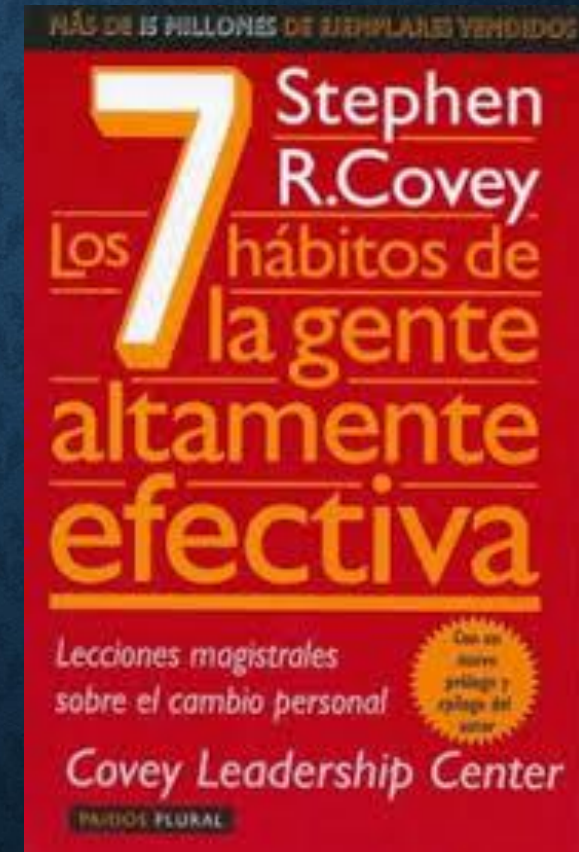
## ➤ **Mejora continua**

- *Estar mejorando los procesos, los equipos y los entornos*



# MATRIZ DE EISENHOWER

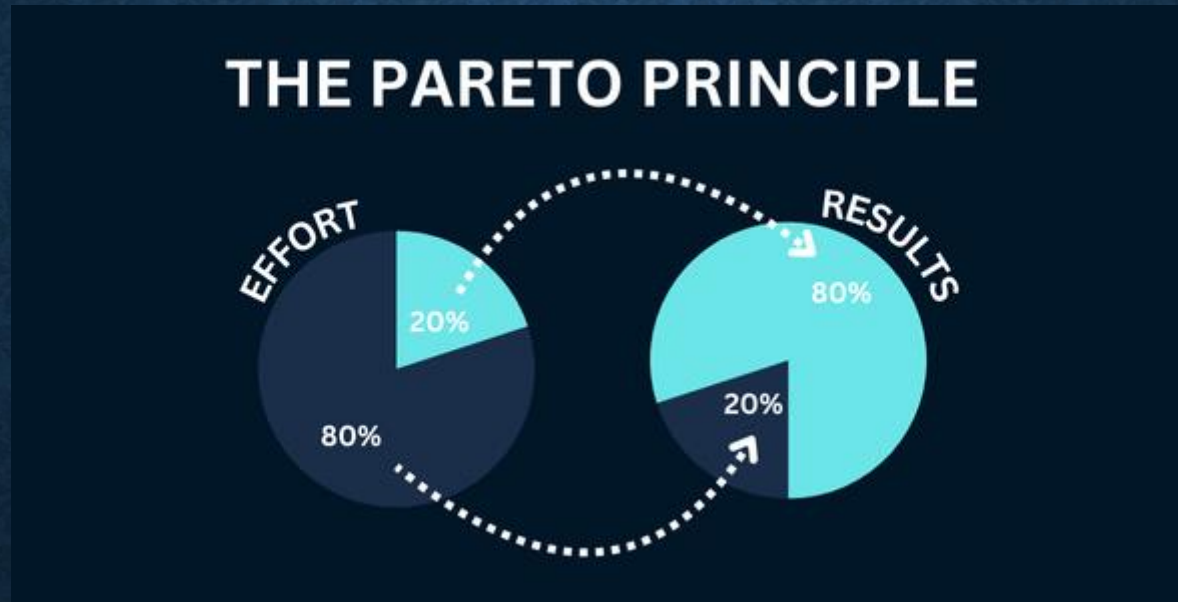
- Popularizada en el libro “Los 7 hábitos de la gente altamente efectiva” de Stephen Covey en 1989



# La matriz de Eisenhower



# REGLA DE PARETO 80/20



Formulada por el economista Vilfredo Pareto en 1896 que observo que el 80% de las tierras en Italia eran propiedad del 20% de la población y llego a concluir que el 20% de las causas genera el 80% de los efectos

El 80% del valor de un sistema de software reside en el 20% de sus funcionalidades

# TECNICA MOSCOW

- **MoSCoW**

- **Must have (Debe tener):** funcionalidades imprescindibles e indiscutibles. Sin ellas el product no funciona (conforman el MVP)
- **Should have (Deberia tener):** Funcionalidades altamente importantes pero no criticas. Sin ellas se reduce eficiencia
- **Could have (Podria tener):** Elementos deseables para confort, estetica o usabilidad
- **Won't have (No se incluyen por ahora):** fuera del alcance por ahora

Not like this....



1



2



3



4

---

Like this!



1



2



3



4



5