



# Análisis y Metodología de Sistemas

Profesor: Darío Tjor  
email: [dario.tjor@ort.edu.ar](mailto:dario.tjor@ort.edu.ar)

# El origen de las Metodologías Ágiles

- Las metodologías ágiles se empezaron a aplicar en la década de los 90 en el sector del software y las nuevas tecnologías debido a la insatisfacción de los trabajadores con los resultados de los modelos tradicionales de gestión.
- En febrero de 2001 se reunieron en Utah (EEUU) un grupo de diecisiete profesionales reconocidos del desarrollo de software con el objetivo de determinar los valores y principios que les permitirían a los equipos desarrollar software de forma más rápida y responder mejor a los cambios que pudieran surgir a lo largo de un proyecto de desarrollo.
- En esta reunión se creó la Agile Alliance, una organización sin fines de lucro cuyo objetivo es el de promover los valores y principios de la filosofía ágil y ayudar a las organizaciones en su adopción. La piedra angular del movimiento ágil es conocida como Manifiesto Ágil ([Agile Manifesto](#)).

# El Manifiesto Ágil



Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

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

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

# El Manifiesto Ágil - Principios

1. Nuestra mayor prioridad es satisfacer al cliente a través de entregas tempranas y frecuentes de software con valor.
2. Aceptar el cambio incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan los cambios para darle al cliente ventajas competitivas.
3. Entregar software funcionando en forma frecuente, desde un par de semanas a un par de meses, prefiriendo el periodo de tiempo más corto.
4. Expertos del negocio y desarrolladores deben trabajar juntos diariamente durante la ejecución del proyecto.
5. Construir proyectos en torno a personas motivadas, generándoles el ambiente necesario, atendiendo sus necesidades y confiando en que ellos van a poder hacer el trabajo.
6. La manera más eficiente y efectiva de compartir la información dentro de un equipo de desarrollo es la conversación cara a cara.
7. El software funcionando es la principal métrica de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los sponsors, desarrolladores y usuarios deben poder mantener un ritmo constante indefinidamente.
9. La atención continua a la excelencia técnica y buenos diseños incrementan la agilidad.
10. La simplicidad –el arte de maximizar la cantidad de trabajo no hecho- es esencial.
11. Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares, el equipo reflexiona acerca de cómo convertirse en más efectivos, luego mejora y ajusta su comportamiento adecuadamente.

[Doce principios del software Ágil](#)

# Metodologías Ágiles

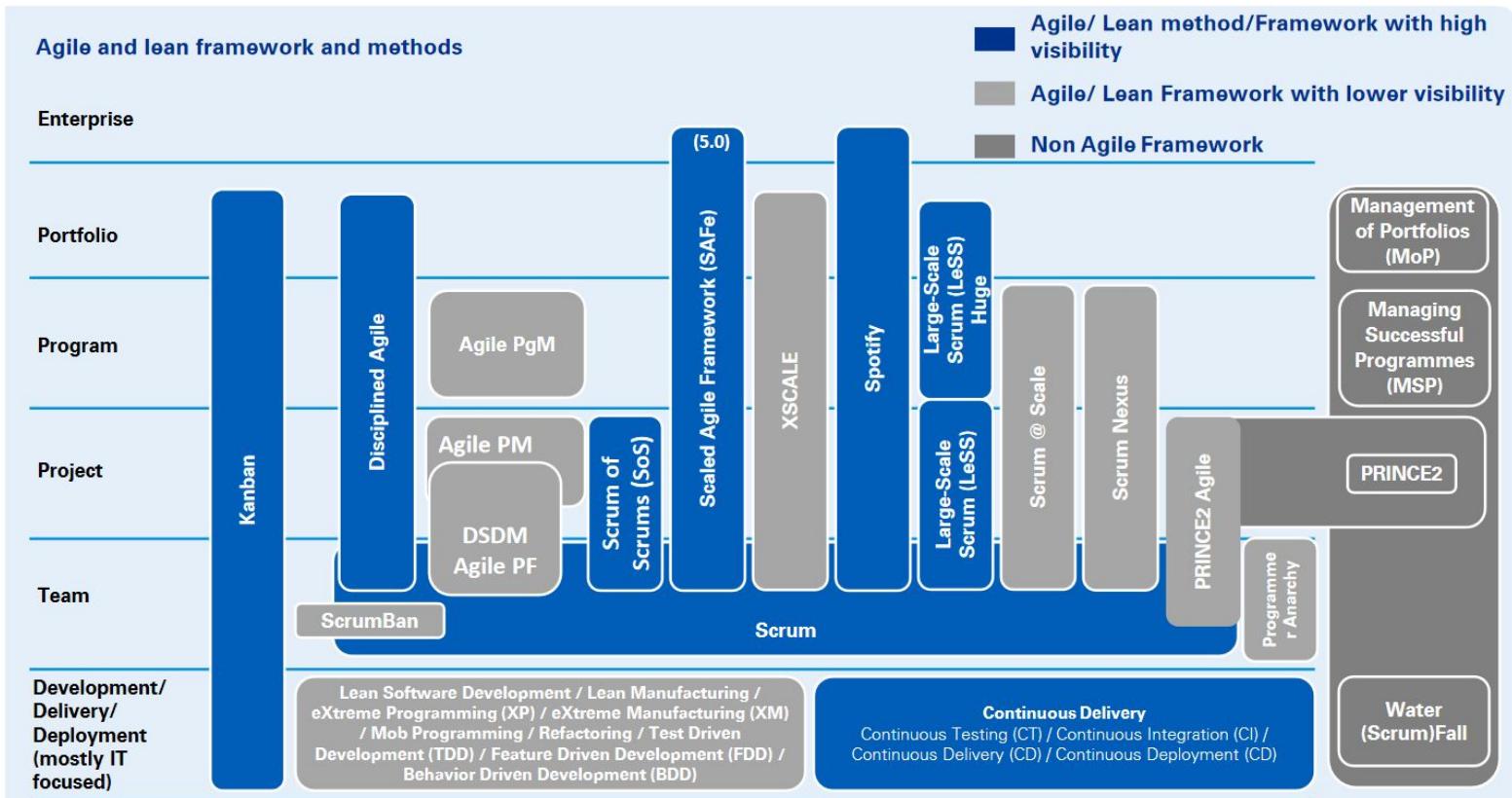
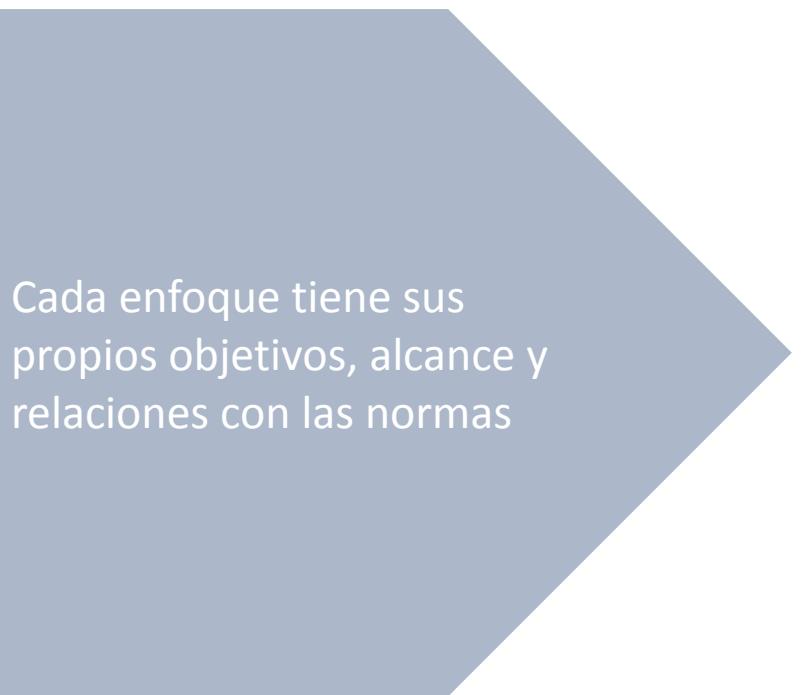


Figure 20: inspired by 'Agile PMG' by Miroslaw Dabrowski, 2015. Expanded based on KPMG insights and market analysis

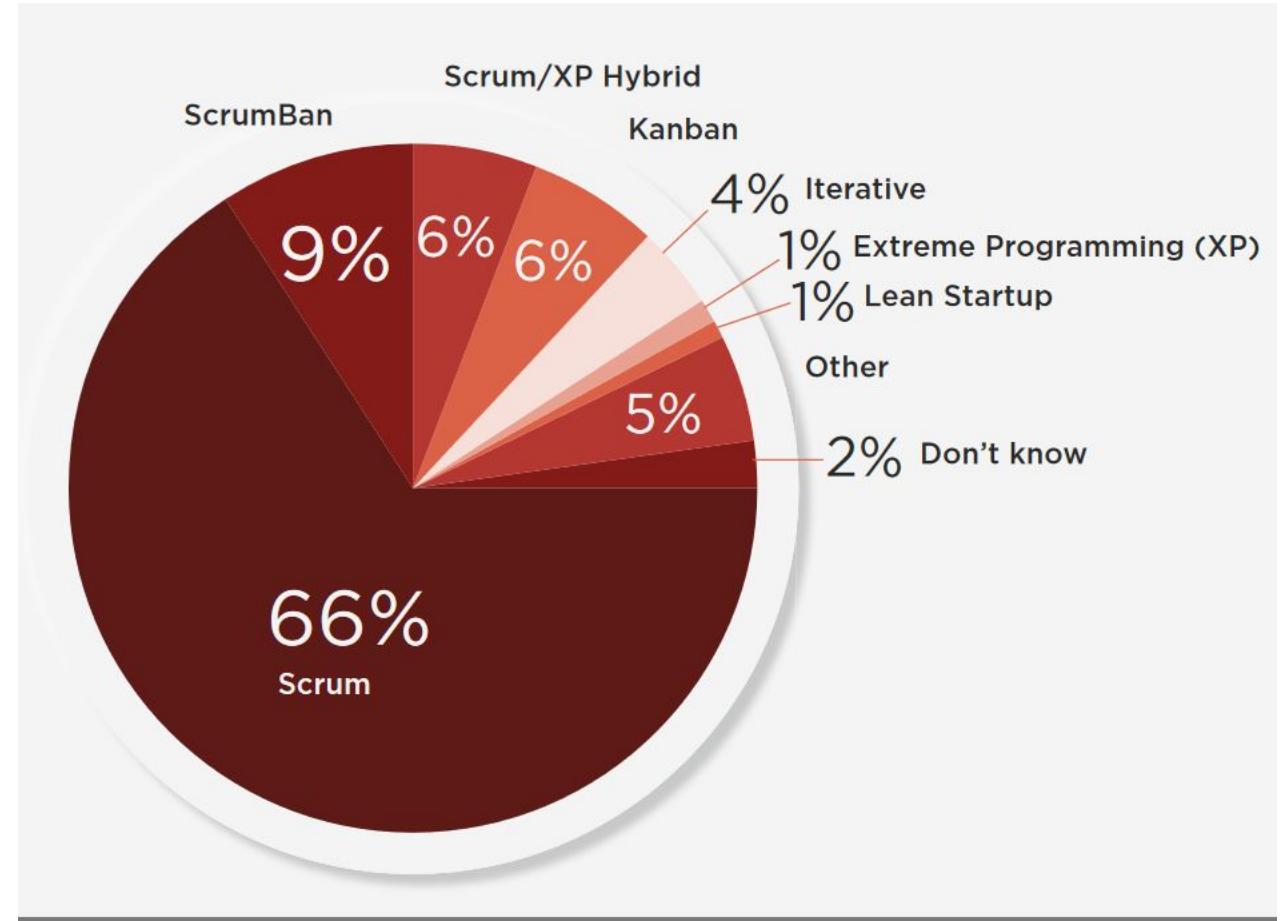
Source: Inspired by Agile PMG by Miroslaw Dabrowski , 2015. Expanded based on KPMG insights and market analysis

# Técnicas ágiles y madurez

A lo largo de los años, la encuesta State of Agile determina "prácticas utilizadas" como un indicio de la madurez de la adopción ágil.

Más notablemente, se ha visto el uso de Scrum aumentar del 40% de los encuestados en la primera encuesta (2006) al 66% en el más reciente (2021)

Otros enfoques como el eXtreme Programming (XP) dejó de utilizarse en casi una cuarta parte de los encuestados a menos del 1% en la actualidad.



Fuente: [State Of Agile Survey – Julio 2021](#)

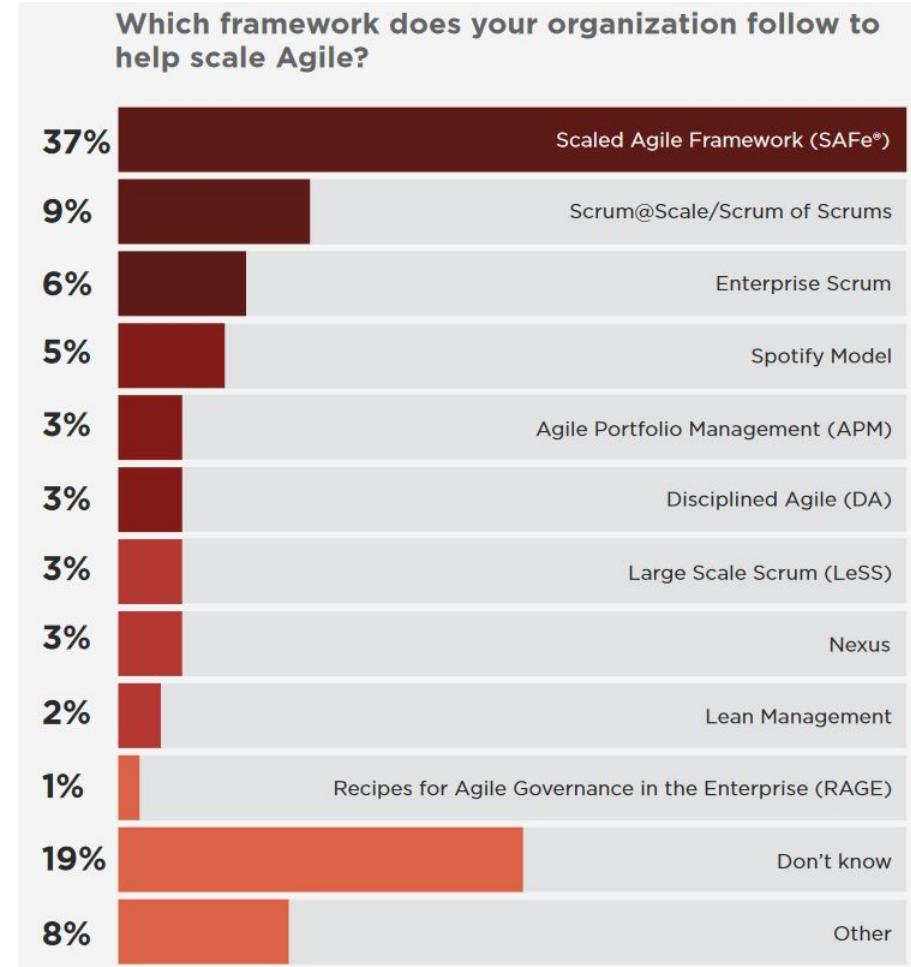
# Scaling Agile

Durante los últimos años, ha ido aumentando la conciencia de las oportunidades y desafíos que ofrece el escalado de prácticas ágiles en toda la organización.

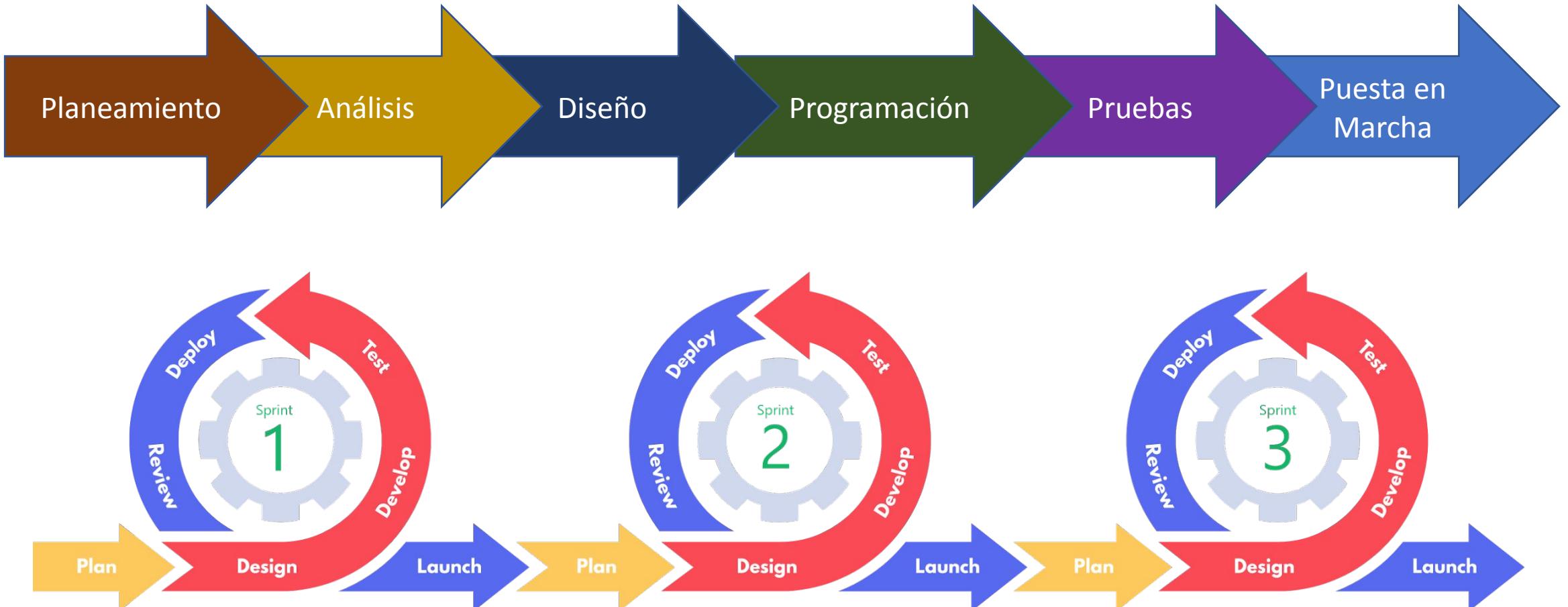
Inicialmente, escalando ágil se abordó a través de un Enfoque: "Scrum of Scrums".

Durante las últimas cinco encuestas, el uso de SAFe® ha crecido significativamente para convertirse en el enfoque dominante, en uso por más de un tercio de los encuestados.

Fuente: [State Of Agile Survey – Julio 2021](#)



# Metodologías Tradicional vs Ágiles



# Cynefin Framework



En un mundo en constante cambio, abordar la complejidad se vuelve esencial para lograr el éxito en nuestros proyectos y decisiones estratégicas. Es aquí donde el modelo Cynefin se convierte en tu guía ofreciendo un enfoque claro y efectivo para navegar por la incertidumbre.

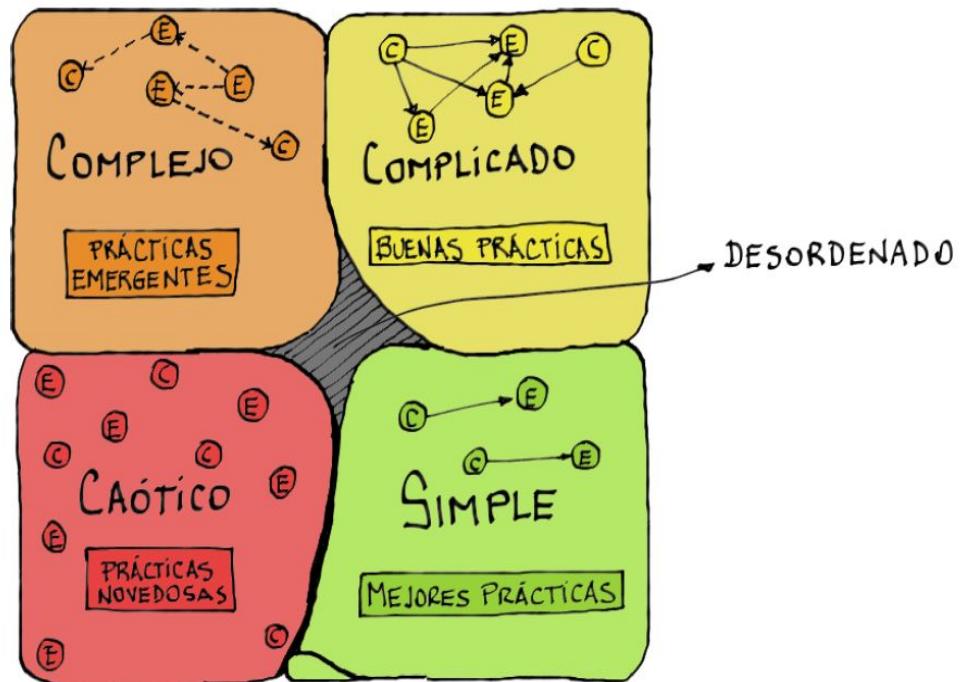
**¿Qué es el modelo Cynefin?** Es un marco conceptual que nos ayuda a comprender y actuar en diferentes dominios de complejidad.

Se basa en tres principios clave: adaptabilidad, auto-organización y retroalimentación continua. Estos principios nos brindan las herramientas necesarias para enfrentar los desafíos de entornos dinámicos y volátiles.

# Cynefin Framework - cont.

Cynefin es una palabra de origen Galés.  
Significa hábitat, refugio, estar familiarizado con, en sí no tiene una traducción exacta al inglés o al español.

Es un marco de trabajo utilizado para facilitar la toma de decisiones, otorgando a quienes toman decisiones cinco dominios, en donde pueden clasificar cualquier problema o necesidad que tengan que resolver y cuál es, según este enfoque, la manera más eficiente de responder a cada una de ellas.



Fuente:  
Proyectos Ágiles con Scrum  
Kleer

# Dominio Simple

- Se opera con problemáticas simples.
- Es muy fácil identificar las causas y sus efectos.
- La respuesta correcta es clara, conocida por todos e indiscutible.
- Existen las mejores prácticas, soluciones conocidas para problemas conocidos.
- Los procesos más eficientes, son aquellos que especifican una serie lógica de pasos y se ejecutan de manera repetitiva, una y otra vez.
- Detectar, Clasificar y Responder
- Si bien Scrum puede funcionar en este contexto, los procesos compuestos por pasos bien definidos son mucho más eficientes.

# Dominio Complicado

- Encontramos problemas complejos, buenas prácticas y perfiles expertos.
- Hay múltiples soluciones correctas para una misma problemática, pero se requiere del involucramiento de expertos para poder identificarlas.
- Una práctica habitual es el mantenimiento de sistemas y soporte técnico.
- Detectar, Analizar y Responder
- Si bien Scrum podría emplearse, no necesariamente sea la forma más eficiente de resolver estas situaciones, donde funcionaría mejor un conjunto de expertos en la materia que releven la situación, investiguen diferentes alternativas y planteen la solución en base a las buenas prácticas.

# Dominio Complejo

- Cuando nos enfrentamos a problemas complejos, los resultados se vuelven más impredecibles.
- No existen ni mejores ni buenas prácticas catalogadas.
- No sabemos con anticipación si una determinada solución va a funcionar. Solo podemos examinar los resultados y adaptarnos.
- Este es el dominio de las prácticas emergentes.
- Experimentar, Detectar y Responder
- El desarrollo de nuevos productos o la incorporación de nuevas características en productos existentes es un contexto complejo en el que Scrum se utiliza mucho para actuar, inspeccionar y adaptar las prácticas emergentes de un equipo de trabajo.

# Dominio Caótico

- Los problemas caóticos requieren una respuesta inmediata.
- Se está en crisis y es necesario actuar de inmediato para restablecer cierto orden.
- Este es el dominio de la improvisación.
- Actuar, Detectar, Responder
- No es un escenario para utilizar Scrum, aquí se debe actuar de inmediato, alguien debe tomar el control y mover la situación fuera del caos.

# Dominio Desordenado

- No sabemos en qué dominio estamos.
- Zona Peligrosa: no podemos medir las situaciones ni determinar la forma de actuar.
- El gran peligro es actuar de manera diferente a la que se necesita para resolver ciertos problemas.
- Todo el esfuerzo debe estar enfocado netamente a salir de este espacio hacia uno mejor identificado, para luego actuar de la manera en que dicho dominio lo requiera.

# Las 5 Etapas de duelo de Kübler-Ross

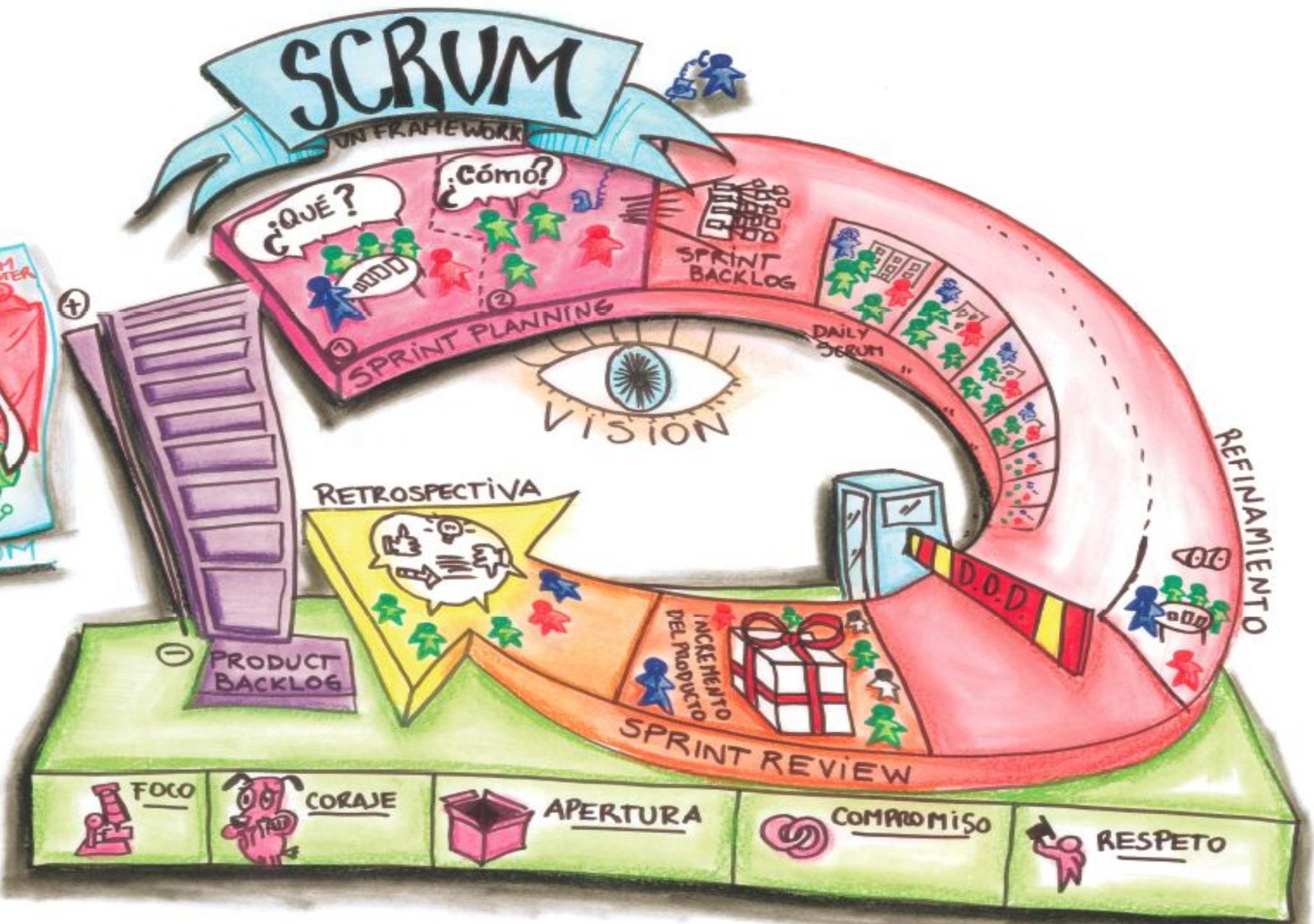


# SCRUM

UN FRAMEWORK



por...  
MARTÍN ALAIMO  
@martinalaimo



# ¿Qué es Scrum?

- Un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente.
- Scrum es:
  - Liviano
  - Fácil de entender
  - Difícil de dominar
- NO es un proceso, una técnica o método definitivo.
- Se basa en la teoría de control de procesos empírica o empirismo.
- Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo.
- Consiste en los equipos Scrum y sus roles, eventos, artefactos y reglas asociadas.

# Los Pilares y Valores de Scrum

## Pilares:

- Transparencia
- Inspección
- Adaptación

## Valores:

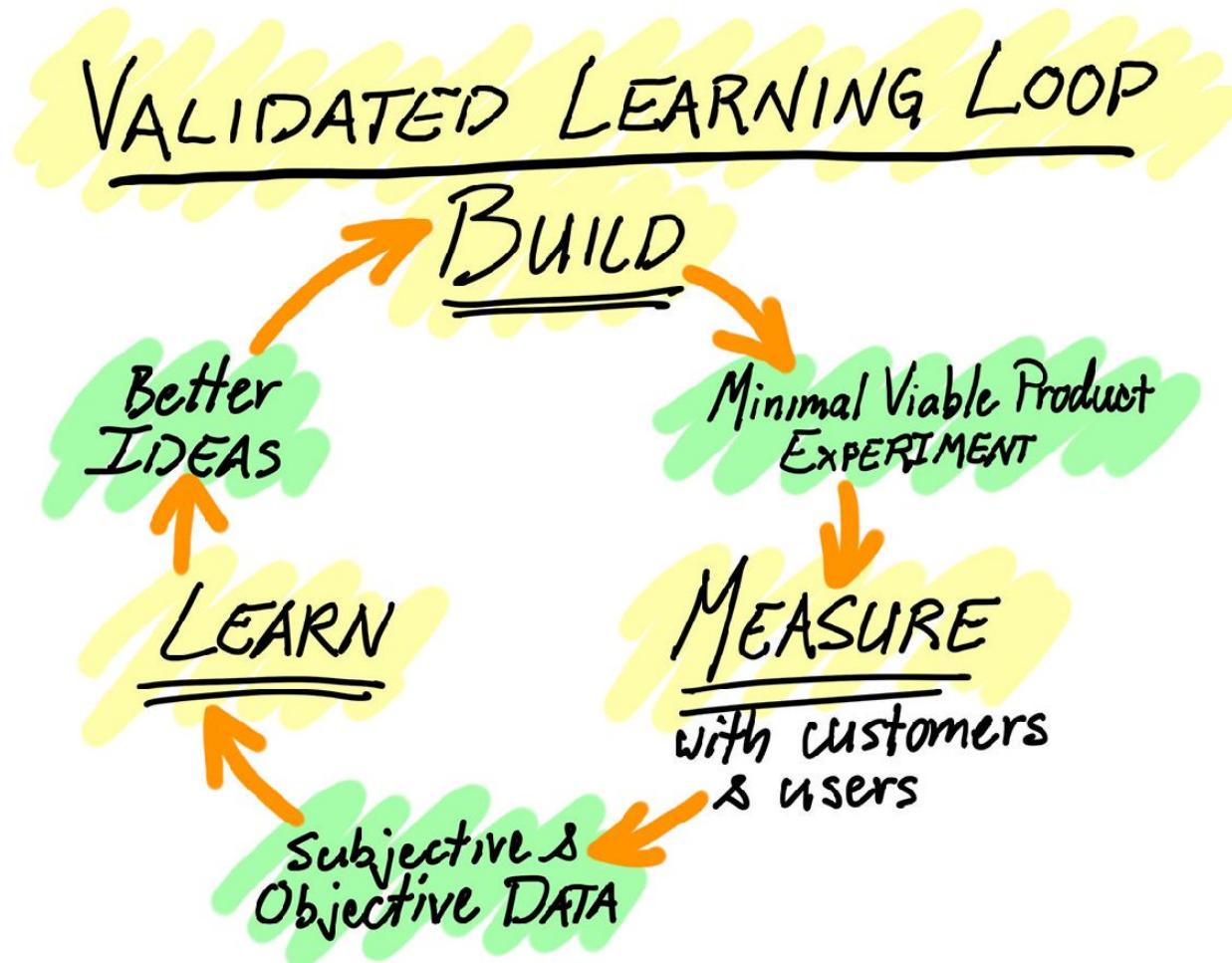
- Foco
- Coraje
- Apertura
- Compromiso
- Respeto



# El Proceso de Scrum



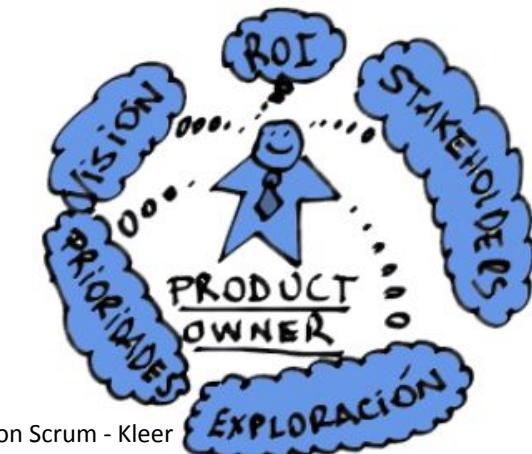
# Aprendizaje Continuo



# Roles de Scrum

## El Product Owner

- Es la persona responsable del éxito del producto desde el punto de vista de los stakeholders.
- Gestiona las expectativas de los stakeholders mediante la comprensión completa de la problemática de negocio y su descomposición hasta llegar al nivel de requerimientos funcionales.
- El Dueño de Producto es la única persona responsable de gestionar el Product Backlog.
- Es una única persona, no un comité.
- Toda la organización debe respetar sus decisiones.
- Tiene un papel de liderazgo.



# Roles de Scrum

## Product Owner - Responsabilidades



- Determinar la **visión** del producto, hacia dónde va el equipo de desarrollo.
- Gestionar las **expectativas** de los stakeholders
- Recolectar los **requerimientos**.
- Determinar y conocer en detalle las **características funcionales** de alto y de bajo nivel.
- Generar y mantener el **release plan**: fechas de entrega y contenidos de cada una.
- Maximizar la rentabilidad (ROI) del producto: **Priorización**.
- Cambiar las prioridades de las características según avanza el proyecto, acompañando así los **cambios** en el negocio.
- Aceptar/rechazar el producto construido durante el Sprint y proveer **feedback** valioso para su evolución.

# Roles de Scrum

## Product Owner – Maximizador de Valor



# Roles de Scrum

## El Equipo de Desarrollo (Dev. Team)



- Consiste en los profesionales que realizan el trabajo de entregar un Incremento de producto “Terminado” que potencialmente se pueda poner en producción al final de cada Sprint.
- Es el único responsable por la construcción y calidad del producto.
- Auto-organizado/auto-gestionado, sin roles asignados externamente.
- Son multifuncionales (incluye miembros con habilidades de testing y a menudo otros no llamados tradicionalmente desarrolladores: analistas de negocio, expertos de dominio, etc.)
- Los Miembros individuales pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo como un todo.
- Intensamente colaborativo.
- Tiene un papel de liderazgo

# Roles de Scrum

## Equipo de Desarrollo

- Tamaño: 5 ~ 9 miembros Ideal:  $7 \pm 2$  miembros .  
El Product Owner & Scrum Master no cuentan en el cálculo del tamaño del equipo.
- Responsabilidades:
  - Proveer las estimaciones.
  - Comprometerse al comienzo de cada Sprint a construir un conjunto determinado de características en el tiempo que dura el mismo.
  - La entrega del producto terminado al finalizar cada Sprint.



Fuente: Proyectos Ágiles con Scrum - Kleer

# Roles de Scrum

## El Scrum Master



- Es el Coach del equipo y es quien lo ayuda a alcanzar su máximo nivel de productividad posible.
- Acompaña al equipo de trabajo en su día a día y garantiza que todos, incluyendo al Product Owner, comprendan y utilicen Scrum de forma correcta.
- Es un líder que está al servicio del Equipo Scrum. (Facilita el proceso de Scrum)
- Ayuda a las personas externas al Equipo Scrum a entender qué interacciones con el Equipo Scrum pueden ser útiles y cuáles no
- Tiene un papel de liderazgo
- Todo incremento debe cumplir con los más altos estándares de calidad definidos dentro de la organización y reflejados en el DOD

# Roles de Scrum

## Scrum Master: Responsabilidades

- Velar por el correcto empleo y evolución de Scrum.
- Facilitar el uso de Scrum a medida que avanza el tiempo.
- Asegurar que el equipo de desarrollo sea multifuncional y eficiente.
- Proteger al equipo de desarrollo de distracciones y trabas externas al proyecto.
- Detectar, monitorear y facilitar la remoción de los impedimentos que puedan surgir con respecto al proyecto y a la metodología.
- Asegurar la cooperación y comunicación dentro del equipo
- Ayuda a resolver los impedimentos.
- Captura datos empíricos para ajustar las previsiones.
- No tiene autoridad en la gestión del equipo (cualquier persona que tenga autoridad sobre el equipo no es, por definición, su ScrumMaster)



Fuente: Proyectos Ágiles con Scrum - Kleer

# Artefactos de Scrum

## El Product Backlog

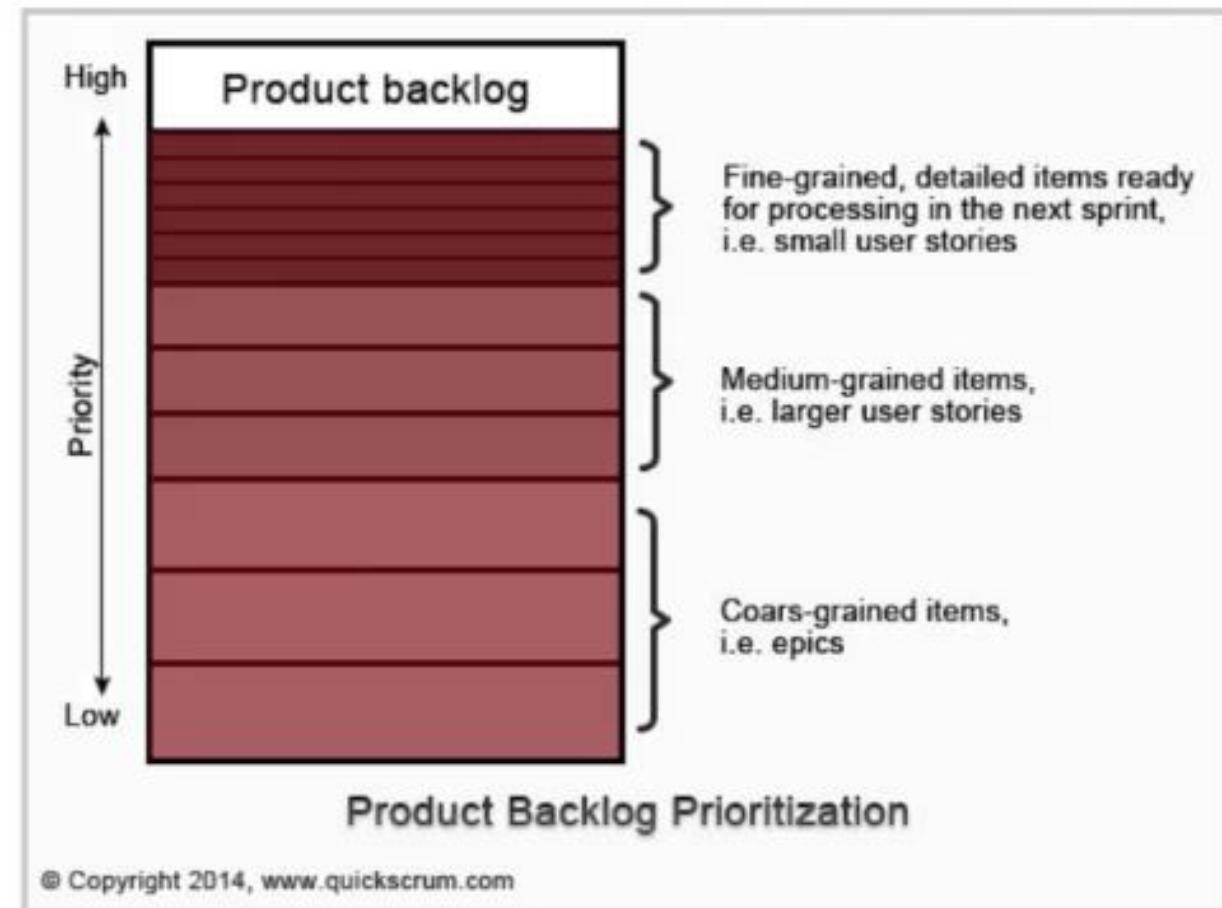


- Es una lista ordenada de todo lo que se conoce que es necesario en el producto.
- Es la única fuente de requisitos para cualquier cambio a realizarse en el producto.
- El Product Owner es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.
- Nunca está completa, es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil.
- Enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras.

# Artefactos de Scrum

## Product Backlog:

- Los elementos de orden más alto son generalmente más claros y detallados que los de menor orden.
- Los elementos (Product Backlog Item) tienen como atributos la descripción, el orden, la estimación y el valor.
- El Equipo de Desarrollo es el responsable de proporcionar todas las estimaciones



# Artefactos de Scrum

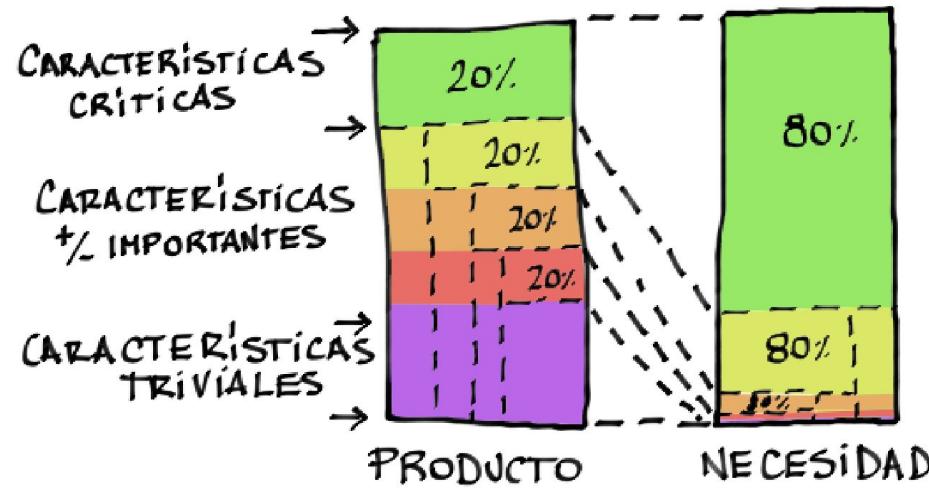
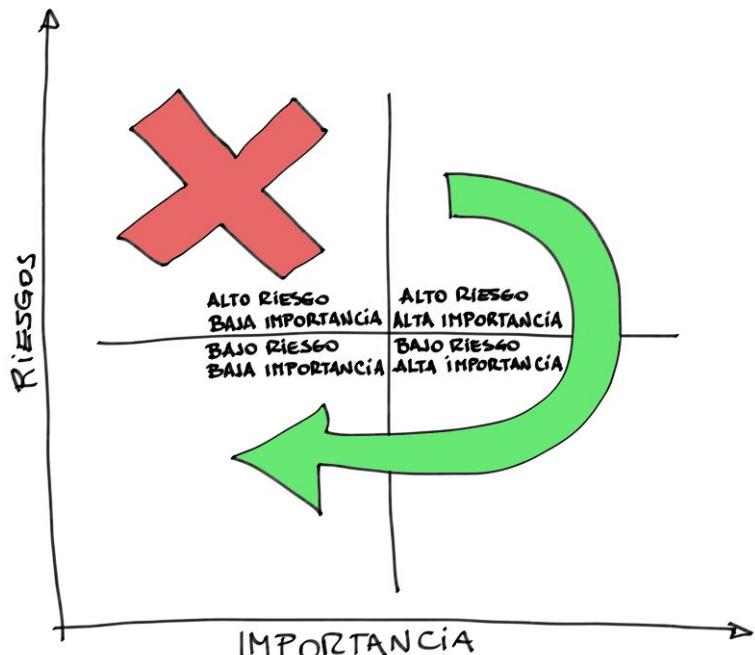
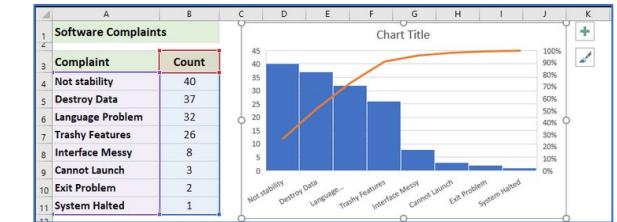
## Product Backlog - Priorización

- Por Importancia:

- Valor del Negocio
- Retorno de la inversión (ROI)  $\square$  ROI = valor de negocio/costo
- Priorización ponderada por riesgos

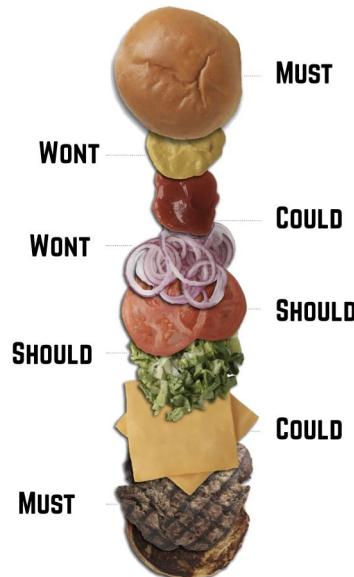
Source: Tertecnic - Ingeniería Predictiva (30-mar-2021)

[El diagrama de Pareto](#)



# Artefactos de Scrum

## Product Backlog - Priorización



Perspectiva de los clientes: Priorización MoSCoW

- Must – Un corazón es un “Debe”. Sin ella, no hay organismo vivo. ¿Qué es un must en tu aplicación?
- Should – Una mano es "debería". Sin ella es difícil. Pero puedes sobrevivir incluso sin una mano.
- Could – El cabello es "podría". Está bien tenerlos, incluso te ves mejor, pero definitivamente sobrevivirás sin ellos.
- Won’t – Desperdicio innecesario. Por cierto, ¿hay algo que "no" en un cuerpo? Incluso el ápendice fue desmitificado.

# Artefactos de Scrum

## Product Backlog - Priorización



### Priorización MoSCoW

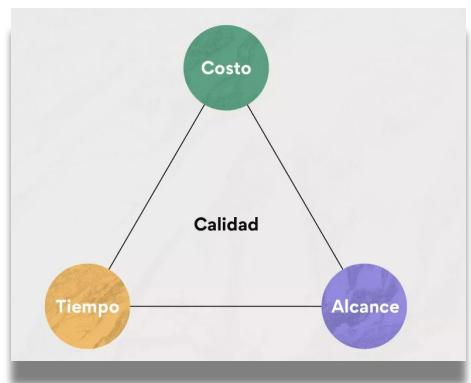
- MUST HAVE: Debe tener - (<80%) – No Negociable
- SHOULD HAVE: Debería haber - (=<20%) – Iniciativas Importantes – No Vitales pero dan mucho valor.
- COULD HAVE: Podría haber - (=<10%) – Encantado que estén. Poco Impacto.
- WILL NOT HAVE: No tendrá que - (100%) . No debe haber.

# Artefactos de Scrum

## - Alcance en los Proyectos Vs La Calidad -

### El triángulo de hierro

- El triángulo de hierro, también conocido como **triángulo de la gestión de proyectos** es un modelo desarrollado por Martin Barnes en 1969 para comprender el balance de las restricciones o limitaciones con las que se encuentran habitualmente los gestores de proyectos.
- Barnes ideó el triángulo de hierro como un modelo con tres pilares que determinan la calidad del proyecto: **alcance, costo y tiempo**.
- El triángulo de hierro demuestra, desde una perspectiva de gestión waterfall o en cascada, cómo están vinculadas estas tres variables: si se cambia una de ellas, las otras dos deben modificarse para mantener el triángulo conectado.
- Si el triángulo se rompe, es decir, si un punto se mueve sin modificar uno o ambos puntos restantes junto con él, la calidad del proyecto se verá afectada.



source: [Triángulo de hierro o de gestión de proyectos: qué es y cómo funciona](#)

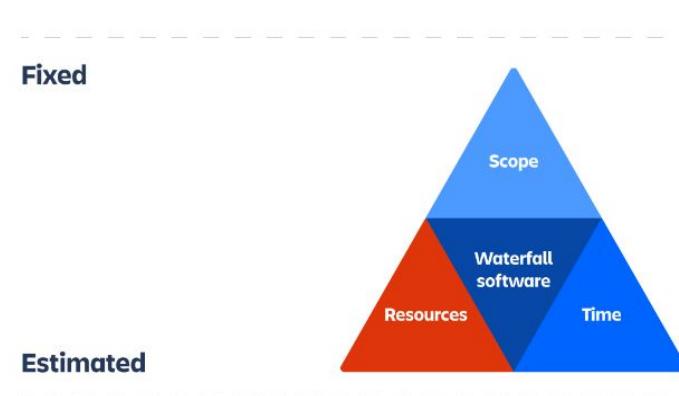
# Artefactos de Scrum

## - Alcance en los Proyectos Vs La Calidad - Cont.

### El triángulo de hierro

- El propósito de la gestión de proyectos de triángulo de hierro es dar a los equipos de productos la información necesaria para hacer compensaciones que ayuden al negocio.
- Las únicas variables con las que pueden jugar son: 1) Tiempo: pueden aceptar una fecha de publicación posterior o 2) Recursos: pueden añadir a algunas personas más al proyecto, lo que aumentará los costes.
- A medida que el desarrollo de software ha evolucionado en el siglo XXI, la necesidad de una mejor colaboración y la capacidad de responder rápidamente a los comentarios de los clientes se ha hecho fundamental, y para ello se creó la metodología ágil.

source: [Gestión de proyectos del triángulo de hierro y la metodología ágil](#)



# Artefactos de Scrum

## - Alcance en los Proyectos Vs La Calidad - Cont.



### Asignación del triángulo de hierro a la metodología ágil

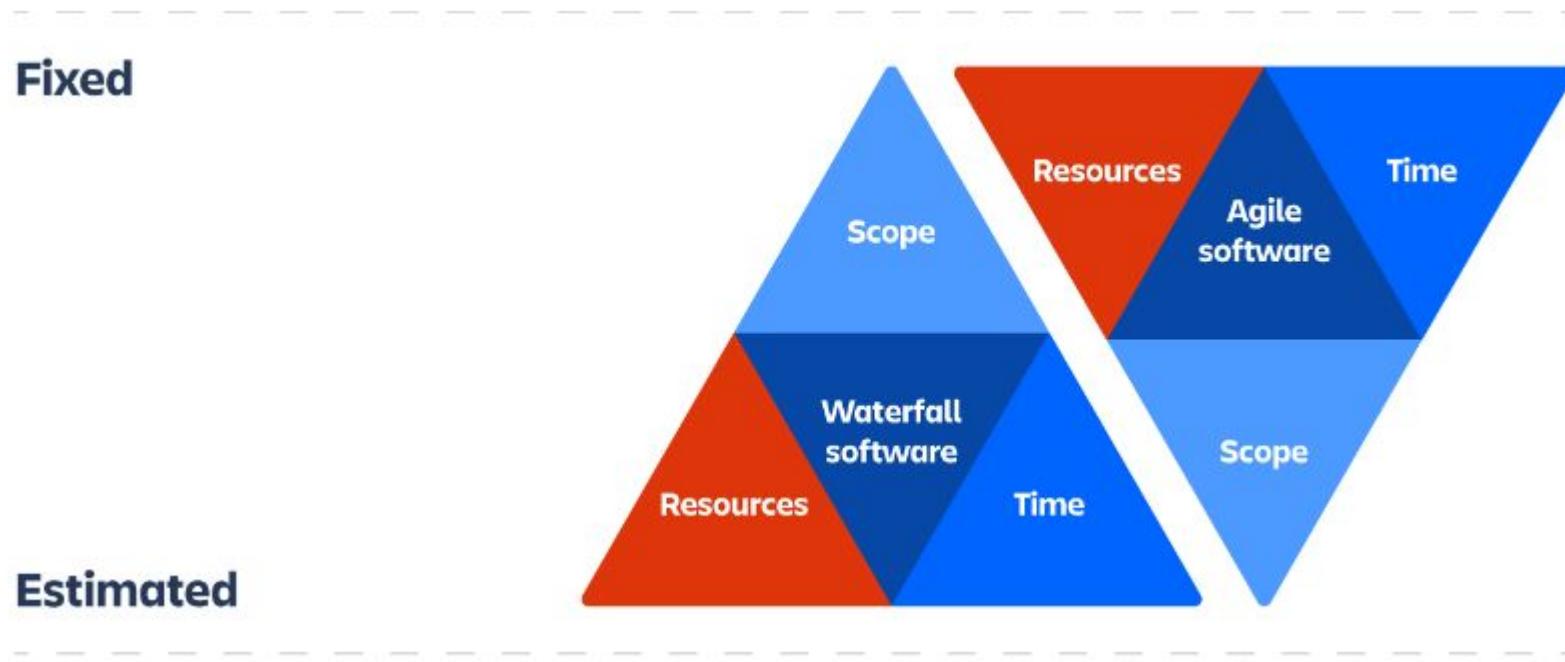
- Lo importante es recordar la diferencia entre lo que es fijo y lo que es estimado.
- A diferencia del desarrollo en cascada, los proyectos ágiles tienen una **planificación y recursos fijos**, mientras que **el alcance varía**.
- Aunque **en el desarrollo ágil el alcance de un proyecto puede cambiar**, los equipos se comprometen a iteraciones de trabajo fijas: sprints, si utilizas un marco scrum, y límites de trabajo en curso, si utilizas un marco kanban.
- Al mantener a los mismos equipos para un producto o proyecto, estos se vuelven más eficaces gracias a la confianza y la continuidad desarrolladas.
- La idea de alcance es la misma en el desarrollo ágil: qué software **compilar** y **entregar**. Sin embargo, la metodología ágil se centra en los **requisitos de alto nivel** en lugar de tratar de incluir requisitos profundos y detallados por adelantado

source: [Gestión de proyectos del triángulo de hierro y la metodología ágil](#)

# Artefactos de Scrum

## - Alcance en los Proyectos Vs La Calidad - Cont.

### Asignación del triángulo de hierro a la metodología ágil



# Artefactos de Scrum

## - Alcance en los Proyectos Vs La Calidad - Cont.



### Asignación del triángulo de hierro a la metodología ágil

- A medida que los proyectos se hacen más grandes, se necesitan más equipos y el recuadro de tiempo se alarga.
- La noción de fijar los recursos y el tiempo, aunque el alcance varíe, no es un enfoque válido para todos los proyectos ágiles.
- La **planificación ágil a largo plazo** requiere un triángulo de hierro **más flexible** que permita a los equipos planificar con antelación y garantice que se están cumpliendo los objetivos de negocio.
- En Lean Startup y en la noción de producto viable mínimo (MVP, del inglés "Minimum Viable Product"). Un MVP, por definición, es un pequeño conjunto de funciones (alcance) que ofrece valor al cliente y para llegar a ese MVP, los equipos podrían tener que **ceñirse a un alcance fijo** (el número de funciones)
- Como **no** se puede publicar sin determinadas funciones, la fecha de publicación se podría retrasar). **Solo** después de lanzar el MVP, los equipos cambian a un alcance variable.

source: [Gestión de proyectos del triángulo de hierro y la metodología ágil](#)

# Artefactos de Scrum

## Product Backlog - Item (PBI)

- Especifica el qué, más que el cómo de una característica centrada en el cliente.
- A menudo escrita en forma de Historia de Usuario.
- Tiene una definición de “terminado” abarcadora de todo el producto para evitar la deuda técnica.
- Puede tener criterios de aceptación específicos del ítem.
- El esfuerzo es calculado por el equipo, de preferencia en unidades relativas (por ejemplo, puntos de la historia).

# Artefactos de Scrum

## El Sprint Backlog



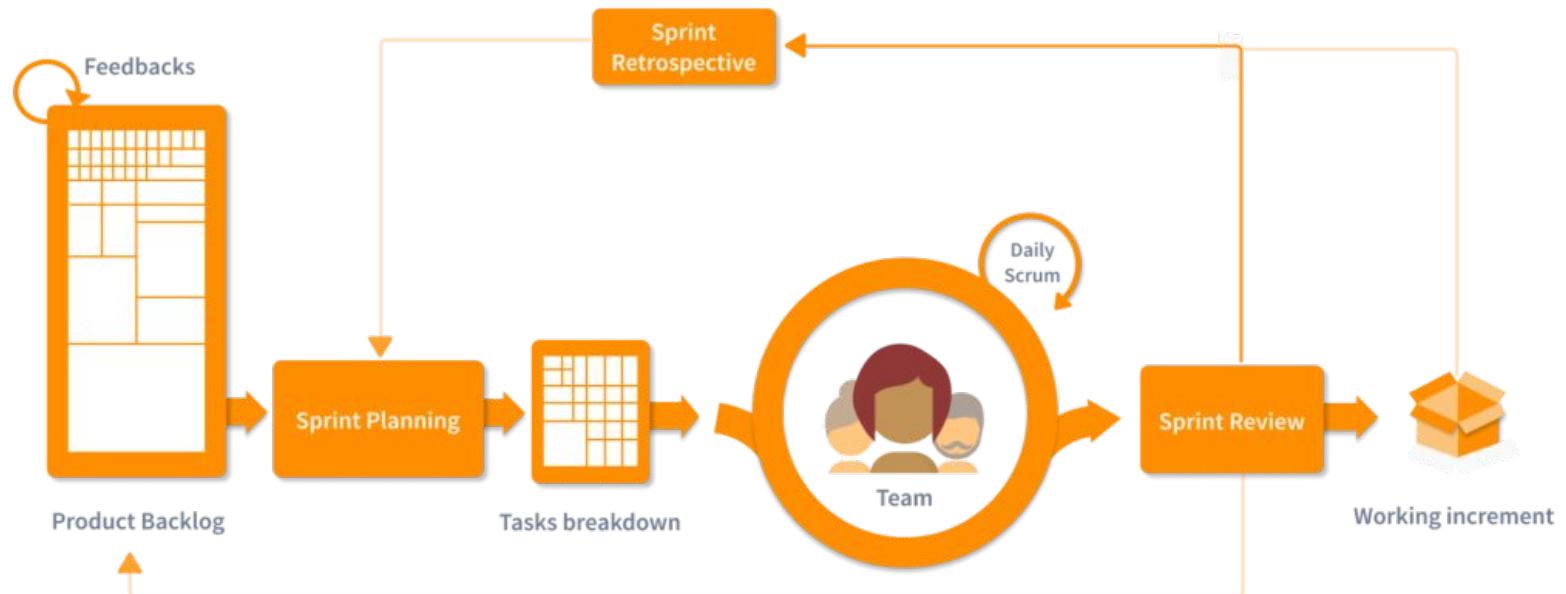
- Es el conjunto de elementos del Product Backlog seleccionados para el Sprint.
- Hace visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint.
- Para asegurar la mejora continua, el Sprint Backlog incluye al menos una mejora de procesos de alta prioridad identificada en la Retrospectiva inmediatamente anterior.
- Solo el Equipo de Desarrollo puede cambiar Sprint Backlog del Sprint durante un Sprint.

# Artefactos de Scrum – Incremento

- Construir software de manera ágil se basa en hacerlo de manera iterativa e incremental.
- Es la suma de todos los elementos del Product Backlog completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores.
- Al final de un Sprint el nuevo Incremento debe estar “Terminado”.
- Es un paso hacia una visión o meta.
- Debe estar en condiciones de utilizarse sin importar si el Product Owner decide liberarlo o no.

# Eventos de Scrum

- Existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum.
- Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima.



# Eventos de Scrum

## El Sprint

- El corazón de Scrum es el Sprint: La Iteración.
- Es un bloque de tiempo (time-box) de un mes o menos.
- Durante el cual se crea un incremento de producto “Terminado” utilizable y potencialmente desplegable.
- Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior.
- Los Sprints contienen: Sprint Planning, Daily Scrums, el trabajo de desarrollo, Sprint Review y Sprint Retrospective.



# Eventos de Scrum

## Sprint



- No se realizan cambios que puedan afectar al Sprint Goal.
- Los objetivos de calidad no disminuyen.
- El alcance puede clarificarse y renegociarse entre el Dueño de Producto y el Equipo de Desarrollo a medida que se va aprendiendo más.
- Limitan el riesgo al costo de un mes calendario.
- Solo el Product Owner tiene la autoridad para cancelar el Sprint.

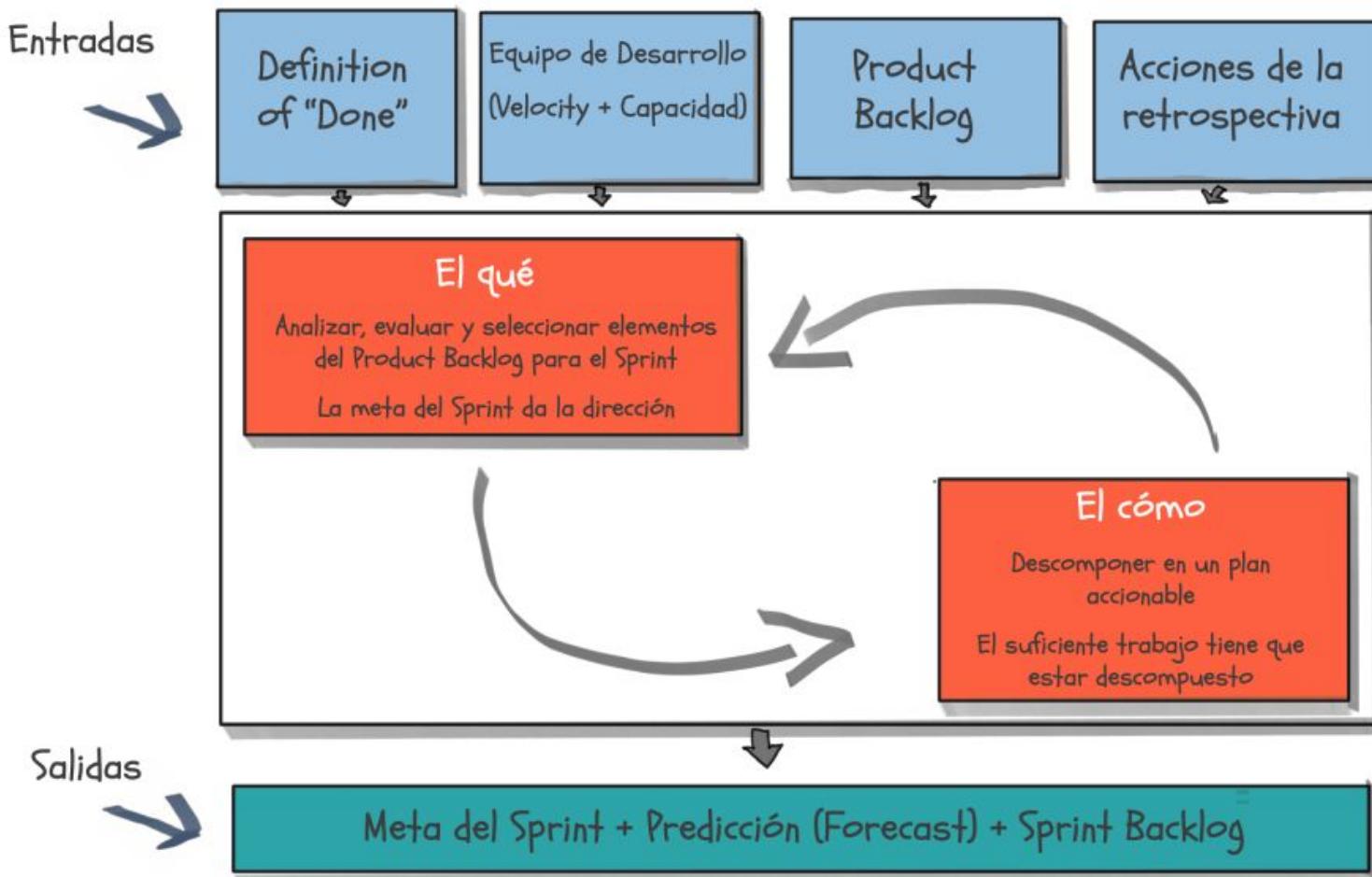
# Eventos de Scrum

## Sprint Planning



- Se planifica el trabajo a realizar durante el Sprint.
- El plan se crea mediante el trabajo colaborativo del Equipo Scrum completo.
- Tiene un máximo de duración de 8 hs. para un Sprint de un mes.
- El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito.
- El número de elementos del Product Backlog seleccionados para el Sprint depende únicamente del Equipo.
- Se define el Sprint Goal.
- Primera parte estratégica: enfocada en “qué”
- Segunda parte táctica: enfocada en el “cómo”

# Sprint Planning



# Eventos de Scrum

## Daily Scrum



- Es una reunión con un bloque de tiempo (time-box) de 15 minutos para el equipo.
- Se lleva a cabo cada día del sprint.
- En el se planea el trabajo para las siguientes 8 horas.
- Optimiza la colaboración y el desempeño del equipo inspeccionando el trabajo avanzado desde el último Daily Scrum.
- Se realiza a la misma hora y en el mismo lugar todos los días para reducir la complejidad.
- Se usa para evaluar el progreso hacia el objetivo.

# Eventos de Scrum – Daily Scrum II

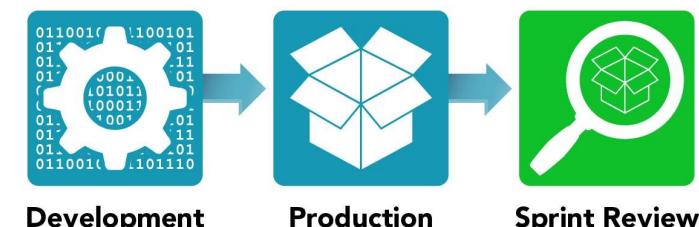
- El Scrum Master se asegura que el Equipo tenga la reunión pero es el equipo el responsable de dirigir el Daily Scrum
- Es una reunión interna del Equipo. Si otras personas están presentes, el Scrum Master se asegura de que no interrumpan la reunión.
- Es una reunión clave de inspección y adaptación.
- El Equipo de Desarrollo es el encargado de establecer la estructura de la reunión y esta se puede conducir de diferentes maneras.



# Eventos de Scrum

## Sprint Review

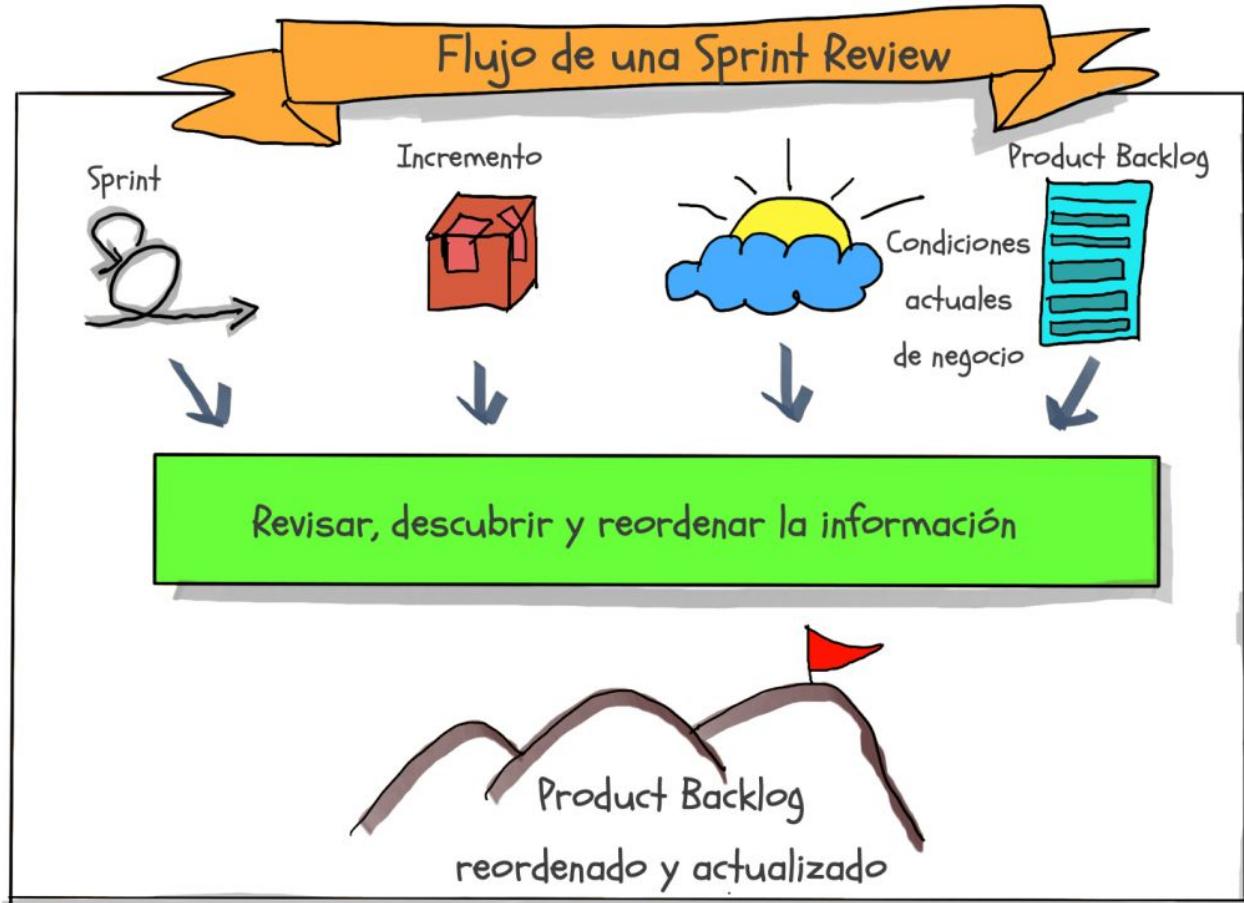
- Se lleva a cabo al final del Sprint.
- El objetivo es inspeccionar el incremento y adaptar el Product Backlog si fuese necesario.
- El equipo y los interesados colaboran acerca de lo que se hizo durante el Sprint.
- Es una reunión informal, no es una reunión de seguimiento.
- Es una reunión de 4hs para Sprints de un mes.
- Asistentes: El Equipo, el Product Owner y los interesados invitados por el Product Owner.
- El product owner explica que elementos del Product Backlog se han terminado y cuales no.
- El equipo hace una demostración del trabajo que se ha terminado y responde a preguntas acerca del incremento.
- El Product Owner habla acerca del Product Backlog en su estado actual y proyecta los objetivos probables y fechas de entrega.



# Eventos de Scrum

## Sprint Review

- El product owner explica que elementos del Product Backlog se han terminado y cuales no.
- El equipo hace una demostración del trabajo que se ha terminado y responde a preguntas acerca del incremento.
- El Product Owner habla acerca del Product Backlog en su estado actual y proyecta los objetivos probables y fechas de entrega.
- La retroalimentación que los stakeholders aporten debe ser ingresada como PBIs en el Product Backlog
- Técnica para Sprint Review: [Sample Sprint Review Agenda & Tips from a Coach](#)



# Eventos de Scrum

## Sprint Retrospective



- Es el corazón de la mejora continua y las prácticas emergentes.
- Tiene lugar después del Sprint Review y antes del siguiente Sprint Planning.
- El equipo reflexiona sobre la forma en la que realizó su trabajo y los acontecimientos que sucedieron en el Sprint.
- Solo participa el Equipo de Desarrollo y el Scrum Master para poder expresarse libremente, sin censura ni temores.

# Eventos del Scrum

## Backlog Refinement

- Es el acto de añadir detalle, estimaciones y orden a los elementos del Product Backlog.
- Es un proceso continuo.
- El Equipo Scrum decide cómo y cuándo se hace el refinamiento.
- Consumo no más del 10% de la capacidad del Equipo de Desarrollo.



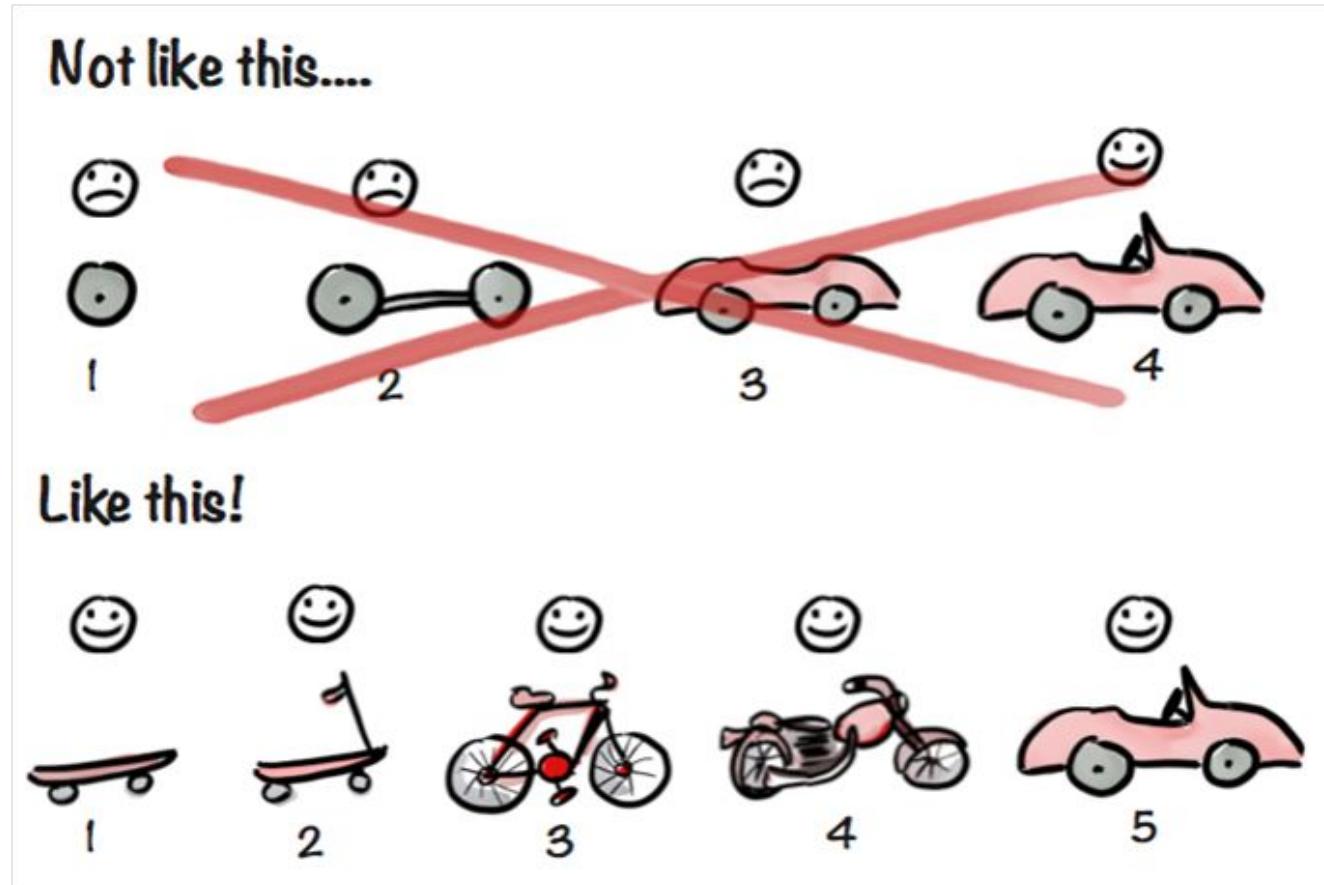
- Facilita la detección de riesgos implícitos.
- El Product Owner es quien la convoca y participa todo el equipo.

# Entrega Iterativa e Incremental

## Producto/Entregable:

El producto se desarrolla por incrementos en el que cada iteración obtiene una versión funcional del producto, de esta forma el sistema se desarrolla poco a poco y obtiene un feedback continuo del usuario

En el grafico, el Skateboard representa el concepto de MVP Minimum Viable Product



Fuente: User Story Mapping: Discover the Whole Story, Build the Right Product™

# Iterativo vs Incremental

- El desarrollo iterativo es cuando los equipos construyen gradualmente las características y funciones, pero no esperan a que cada una de ellas esté completa antes de lanzarlas.

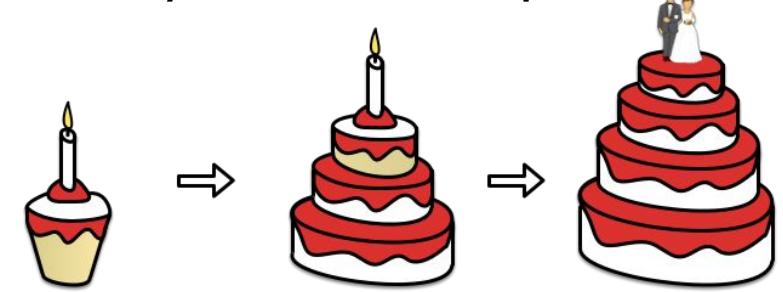


- El desarrollo incremental es un enfoque de desarrollo que divide el producto en segmentos completamente funcionales que se denominan incrementos.



# Minimum Viable Product

- El Minimum Viable Product (MVP) es la versión mínima de un producto, tal que nos permita recolectar la mayor cantidad de información de nuestro mercado y clientes con el menor esfuerzo posible.
- Consiste en hacer foco en las características mínimas y necesarias para que el producto pueda lanzarse al mercado.
- Permite:
  - Evitar crear productos que nadie necesita
  - Maximizar el aprendizaje por dólar invertido
- El MVP es una estrategia de Lean Startup que apunta a acercarnos a nuestros clientes con la menor inversión posible (tiempo/dinero) y con ello determinar si nuestro producto es o no es viable.



# Minimum Marketable Features

- Se define como:

“El conjunto más pequeño posible de funcionalidad que, por si misma, tiene valor en el mercado”



(minimum viable product)



(minimum marketable product)



(product)

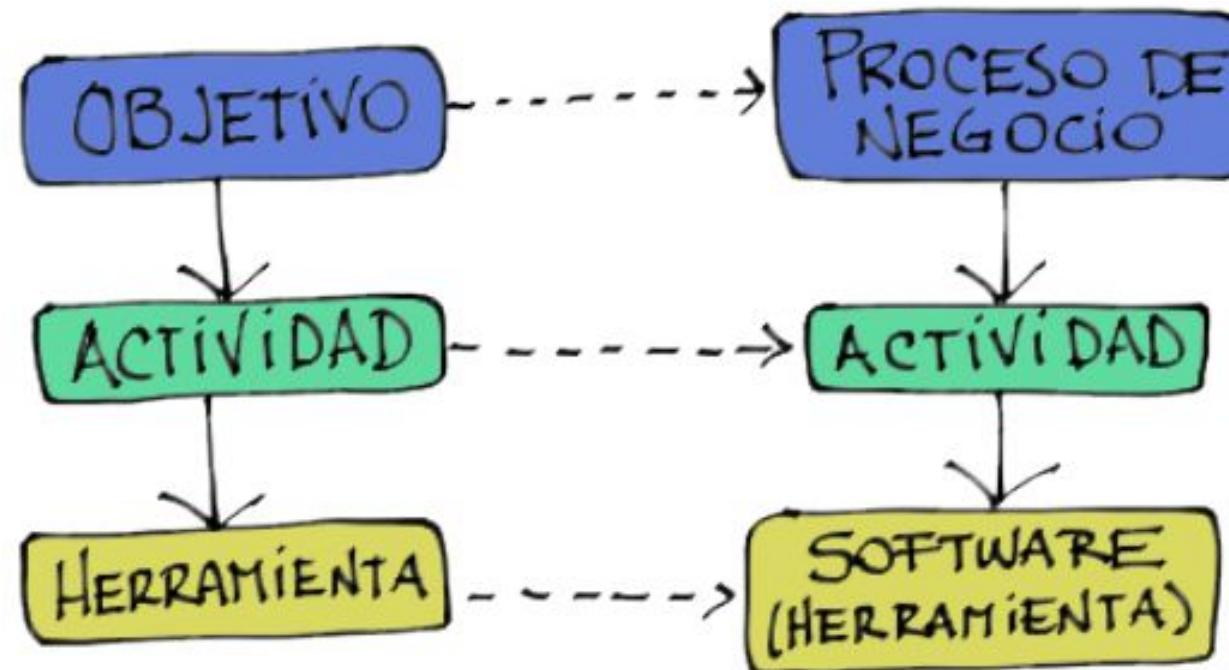
# User Story Mapping

- ¿Qué es User Story Mapping?
- La técnica **User Story Mapping** consiste en armar un mapa visual en el cual podamos planificar, organizar y priorizar nuestro backlog del producto. Se encarga de dejar plasmado el viaje de un usuario en nuestra aplicación, con las tareas y actividades que realizará. Este mapa se arma en conjunto con todo el equipo en el cual nos aseguremos que todos estemos alineados a la hora de planificar nuestro producto.
- Jeff Patton plantea una técnica de Análisis Ágil llamada User Story Mapping
- La teoría del **User Story Mapping** comienza en un nivel “humano” identificando los Objetivos que toda persona persigue y dividiéndolos en Actividades para las cuales deben utilizarse Herramientas, resultando entonces en una jerarquía de:

**Objetivos → Actividades → Herramientas.**

# User Story Mapping

- Haciendo una analogía con las organizaciones, esta jerarquía de Objetivo → Actividad → Herramienta puede traducirse en Proceso de Negocio → Actividad → Software



Jerarquía Operativa

# User Story Mapping

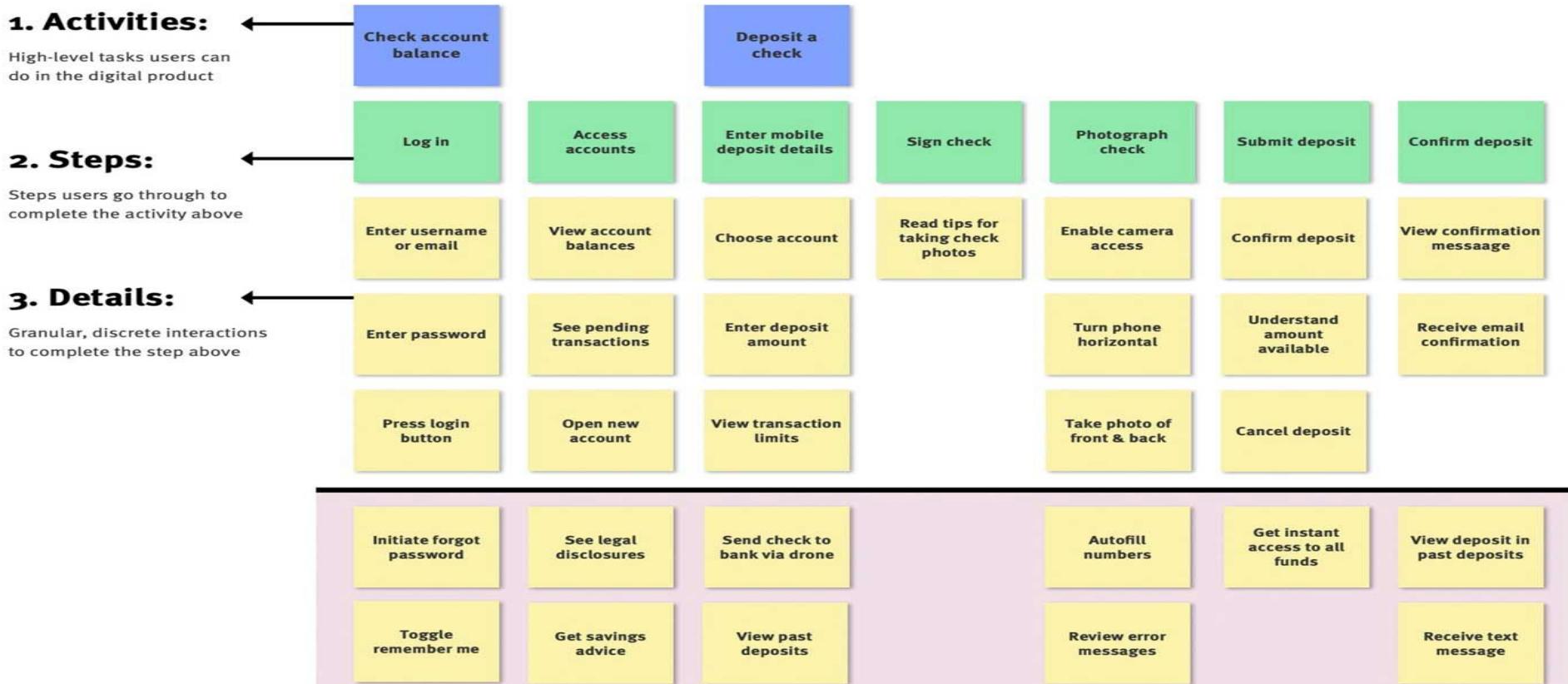
## ¿Cómo armo un User Story Mapping?

- Las **actividades** representan las tareas de alto nivel que los usuarios pretenden completar en el producto digital, por ejemplo, consultar el saldo de la cuenta o depositar un cheque. Dependiendo del tipo de aplicación o sitio web que esté creando, es posible que solo tenga algunas actividades de alto nivel. Estos pueden mostrarse en orden secuencial o en paralelo si existen múltiples rutas para varios tipos de usuarios. La investigación exploratoria sobre las principales tareas de los usuarios debería informar este nivel del mapa.
- Las **tareas** se ubican directamente debajo de las actividades y también se muestran en orden secuencial. Representan las subtareas específicas que los usuarios realizarán en el producto para completar la actividad anterior. Por ejemplo, la actividad Depositar un cheque se puede dividir en los pasos Introducir detalles de depósito móvil, Firmar cheque, Fotografiar cheque, Enviar depósito y Confirmar depósito.
- Los **detalles** son el tercer nivel del story map y describen las interacciones de menor granularidad que el equipo anticipa que experimentarán los usuarios para completar el paso anterior. Por ejemplo, Ingresar nombre de usuario o correo electrónico e Ingresar contraseña aparecen como dos detalles separados debajo del paso Iniciar sesión.

# User Story Mapping

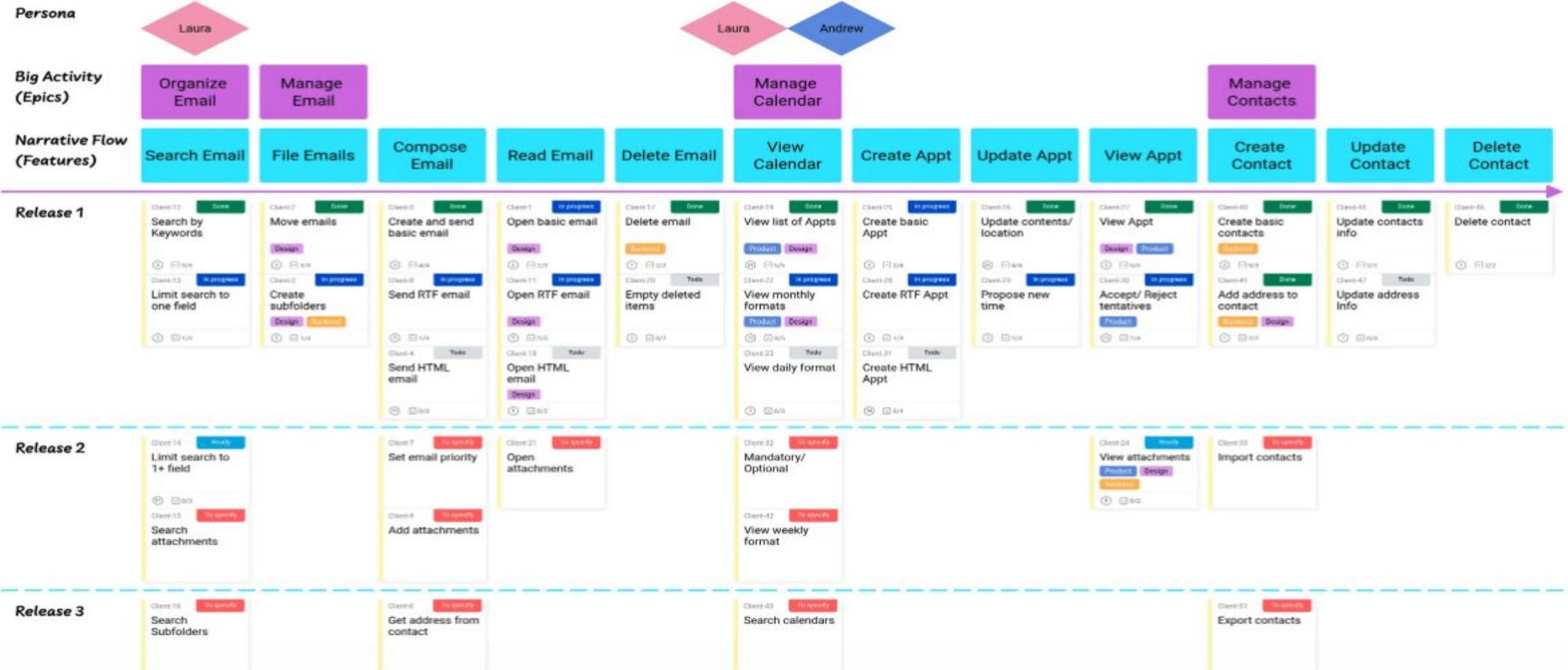
**User-Story Map: Mobile App Feature for Depositing Checks**

NNGROUP.COM NN/g

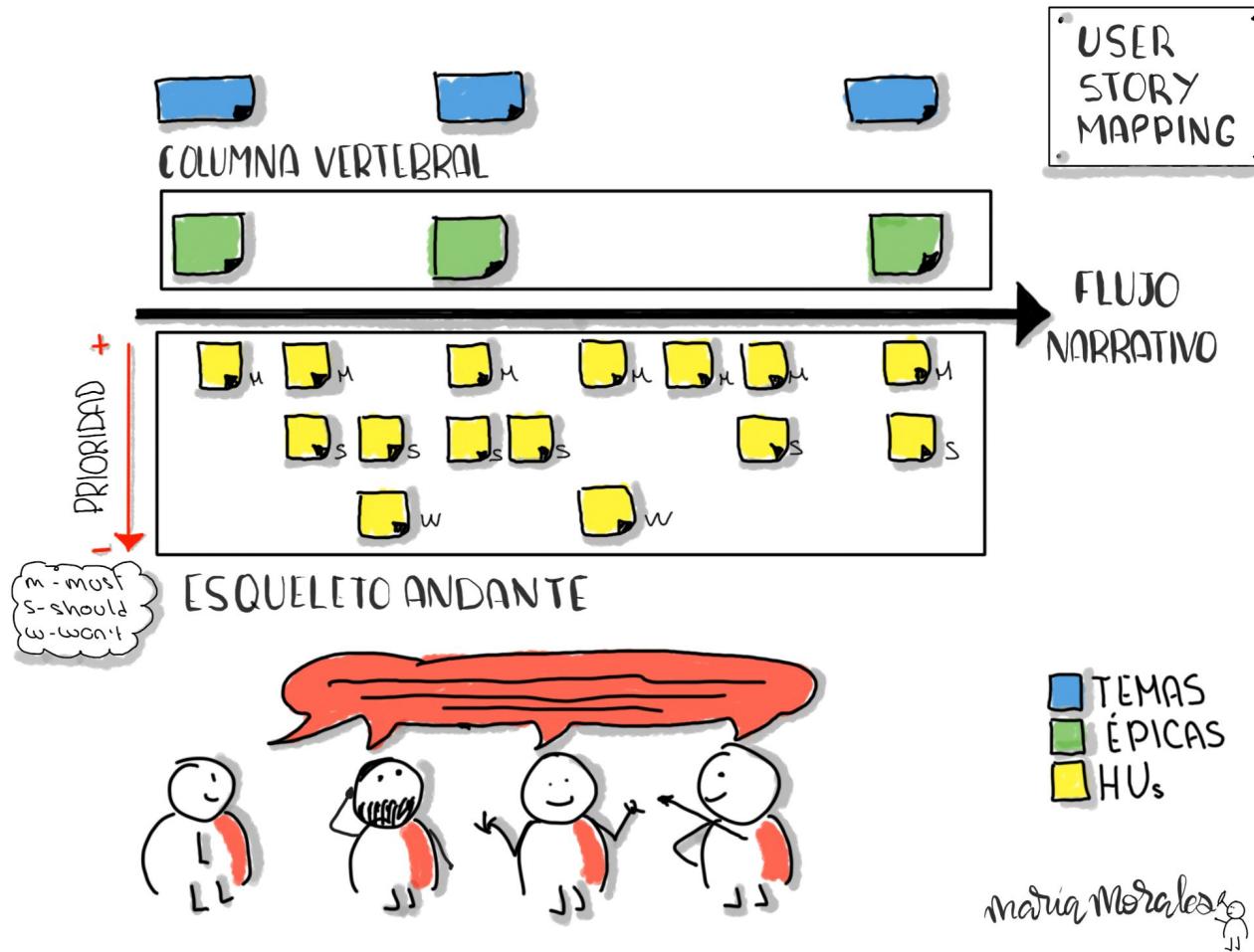


# User Story Mapping

## User Story Mapping



# User Story Mapping



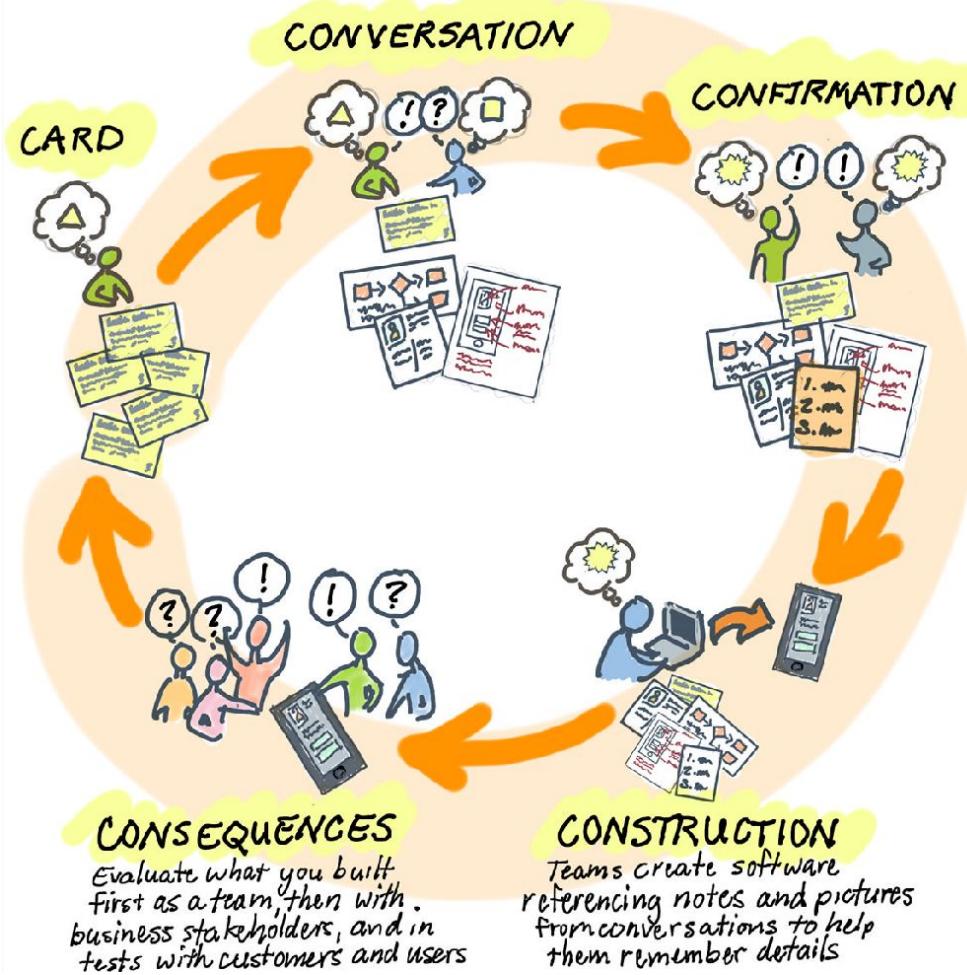
# Historias de Usuario / User Stories

- Las Historias de Usuario surgieron en eXtreme Programming (XP) como una respuesta a una situación habitual en los proyectos de desarrollo de software
- Una Historia de Usuario se compone de 3 elementos, también conocidos como “las tres Cs” de las Historias de Usuario:



- **Card (Ficha)** – Toda historia de usuario debe poder describirse en una ficha de papel pequeña. Si una Historia de Usuario no puede describirse en ese tamaño, es una señal de que estamos traspasando las fronteras y comunicando demasiada información que debería compartirse cara a cara.
- **Conversación** – Toda historia de usuario debe tener una conversación con el Product Owner. Una comunicación cara a cara que intercambia no solo información sino también pensamientos, opiniones y sentimientos.
- **Confirmación** – Toda historia de usuario debe estar lo suficientemente explicada para que el equipo de desarrollo sepa qué es lo que debe construir y qué es lo que el Product Owner espera. Esto se conoce también como Criterios de Aceptación.

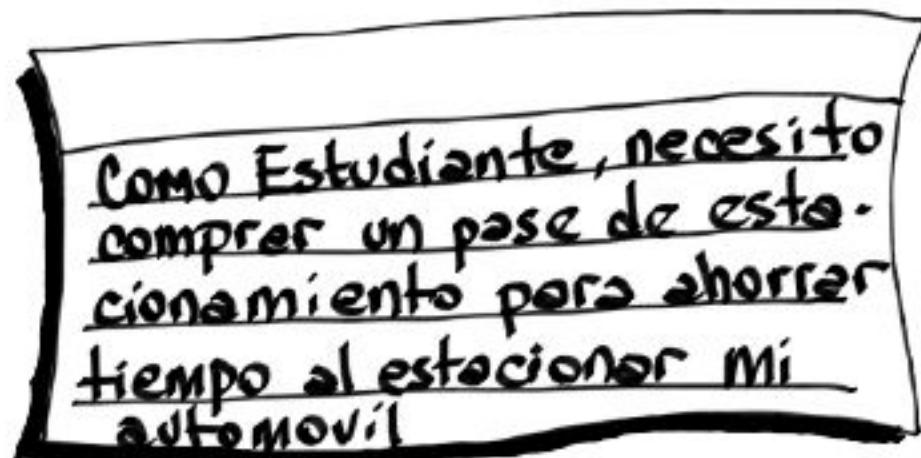
# User Stories – Card Solo el inicio



# User Stories - Redacción

- Mike Cohn sugiere una determinada forma de redactar Historias de Usuario bajo el siguiente formato:

Como (rol) Necesito (funcionalidad) Para (beneficio)



# User Stories - INVEST



- **Independientes:** Las Historias de Usuario deben ser independientes de forma tal que no se superpongan en funcionalidades y que puedan planificarse y desarrollarse en cualquier orden.
- **Negociable:** No es un contrato explícito por el cual se debe entregar todo-o-nada. Por el contrario, el alcance de las Historias (sus criterios de aceptación) podrían ser variables: pueden incrementarse o eliminarse con el correr del desarrollo y en función del feedback del usuario y/o la performance del Equipo.
- **Valorable:** Una Historia de Usuario debe ser Valorable por el Product Owner.

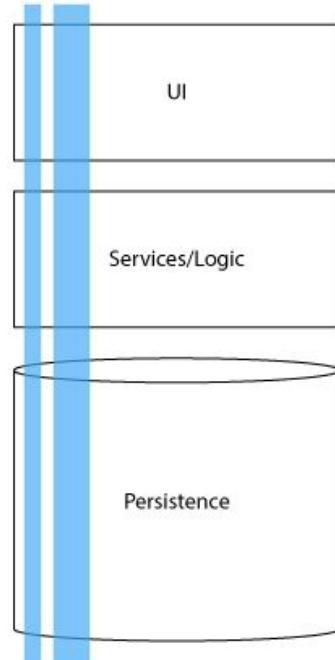
# User Stories - INVEST

- **Estimable:** Mike Cohn, identifica tres razones principales por las cuales una Historia de Usuario no podría estimarse:
  1. Demasiado grande
  2. Falta de conocimiento funcional
  3. Falta de conocimiento técnico
- **Pequeña:** Algunos Equipos fijan el tamaño de una Historia de usuario como no más de dos semanas de una persona. Si bien no es una medida explícita, tener entre 4 y 6 Historias de Usuario por Sprint es una buena señal de tamaño.
- **Verificable:** Se espera que el Product Owner no solo pueda describir la funcionalidad que necesita, sino que también logre verificarla (probarla). Algunos Equipos acostumbran solicitar los criterios de aceptación antes de desarrollar la Historia de Usuario.

# User Stories - División

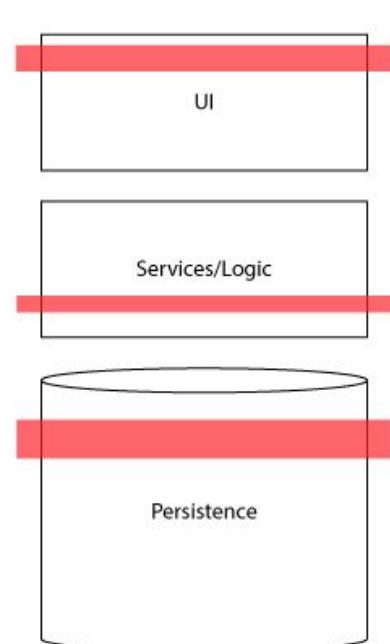
## Vertical Slices

include changes to each architectural layer sufficient to deliver an increment of value



## Horizontal Slices

multiple slices must be completed to deliver an increment of value



## Story

Stories describe something we can deliver and evaluate



## Delivery Tasks

Delivery tasks describe the work we need to accomplish to realize the story



**Big Story → Big Recipe**



**Small Story → Similar recipe, just less of everything**

# Definición de Listo (DOR)

También conocido como Definition of Ready, es el conjunto de características que una Historia de Usuario debe cumplir para que el Equipo de Desarrollo pueda comprometerse a su entrega, es decir, incluirla en un Sprint Backlog.



# Ejemplo DOR

## Definition of Ready: Ejemplos

Como referencia podríamos tener en cuenta los criterios más comunes que se utilizan en la definición del DoR, aunque habrá que tener en cuenta la particularización al desarrollo de trabajo en el equipo ágil donde se va a tener en cuenta.

- La tarea debe tener una estimación de complejidad (puntos de historia).
- El detalle funcional, detalle técnico y los casos de prueba de cada tarea deben ser claros para que sean entendibles por cualquier miembro del equipo.
- La tarea no debe tener bloqueos que impidan su ejecución.
- Las dependencias deben estar resueltas.
- La tarea debe poder validarse y verificarse dentro del Sprint.

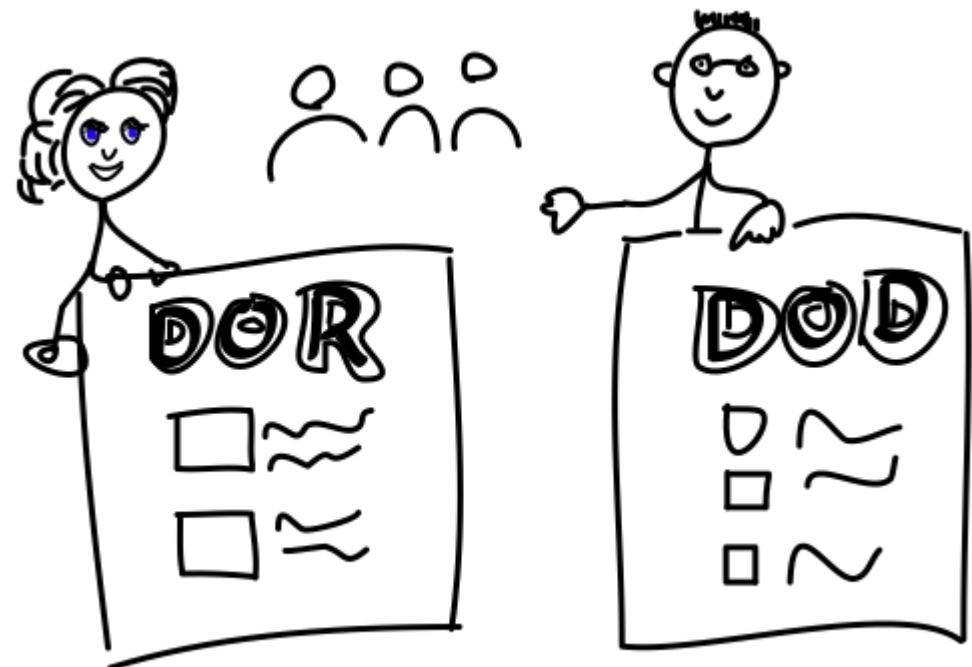
## Consecuencias de no tener en cuenta el Definition of Ready

Si incluimos tareas en el Sprint que no cumplen con el DoR podemos encontrarnos con cosas como:

- Tiempo empleado en la tarea tirado a la basura porque no puede terminarse la tarea dentro del sprint.
- Desorientación en el equipo a la hora de abordar tareas que no cumplen los requisitos iniciales del **DoR**.
- Desconfianza en los clientes al poner en riesgo el valor incremental del producto al finalizar el sprint.

# Definición de Terminado (DOD)

- También conocido como Definition of Done, es el conjunto de características que una Historia de Usuario debe cumplir para que el equipo de desarrollo pueda determinar si ha terminado de trabajar en ella.



# Ejemplo DOD

## ¿Quién actualiza el Definition of Done y cada cuánto?

El DoD puede cambiar con el tiempo y se actualiza siempre que sea necesario, pero nunca en medio de un Sprint. Generalmente se utiliza el Sprint Retrospective para actualizar el Definition of Done.

## Posibles ejemplos de DoD

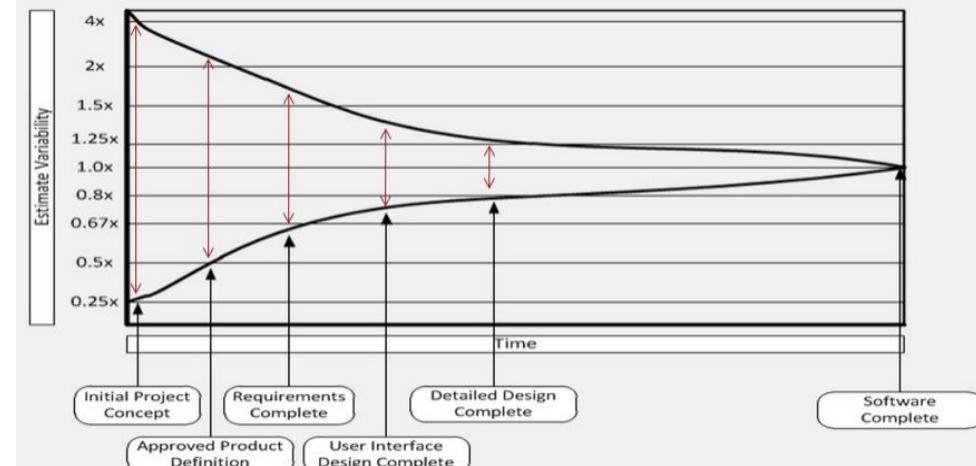
- Todas las pruebas unitarias y funcionales son correctas.
- Todos los criterios de aceptación se cumplen.
- OK del equipo: UX, desarrollador, Product Owner, etc.
- Pruebas en dispositivos/navegadores pasada.
- Pruebas de rendimiento pasadas.
- Se han corregido todos los bugs.
- Entorno preparado para la subida a producción.

# Cono de la Incertidumbre

- En la [gestión de proyectos](#), el cono de Incertidumbre describe la evolución de la medida de incertidumbre durante la realización de un proyecto. La incertidumbre no solo se reduce conforme pasa el tiempo, sino que también disminuye su impacto en la [gestión de riesgos](#), especialmente en la [toma de decisiones](#).
- El término "Cono de incertidumbre" es usado en la [Ingeniería de Software](#), cuando los ambientes técnicos y de negocios cambian de manera rápida y repentina. Sin embargo, el concepto, bajo un nombre diferente, es también un principio básico establecido de la Ingeniería de costos.
- La mayoría de los ambientes cambian de manera lenta, tanto que pueden ser considerados como "estáticos" para la duración de un proyecto típico, y, por ende, los métodos tradicionales de gestión de proyectos se enfocan en lograr un entendimiento completo del ambiente a través de un análisis y planeación cuidadosos. Mucho antes de que se hagan inversiones significativas, la incertidumbre es reducida a un nivel en el cual los riesgos pueden ser eliminados o tratados de manera cómoda.

Estimación de variabilidad Vs Tiempo

El cono de Incertidumbre describe la **variabilidad** durante la realización de un proyecto.



La precisión aumenta a lo largo del tiempo asociada a la certeza y detalle.

# Estimaciones en contextos inciertos

- Para mitigar el riesgo de proveer estimaciones incorrectas, en metodologías ágiles se opta por reducir la precisión de las estimaciones en función de cuánto conocimiento se tiene sobre el esfuerzo que se requiere estimar. De esta manera, los “requerimientos” y sus “estimaciones” se categorizan en diferentes niveles de precisión.
- Escalas de PBIs y Estimaciones:
  - Alto Nivel: EPIC (bloque funcional) estimada en Tamaño (T-Shirt) (XS, S, M, L, XL)
  - Nivel Medio: Historia de Usuario (funcionalidad) estimada en Puntos de Historia (Sucesión de Fibonacci) - Estimación Relativa
  - Bajo Nivel: tareas o actividades estimadas en horas, preferiblemente menos de un día.

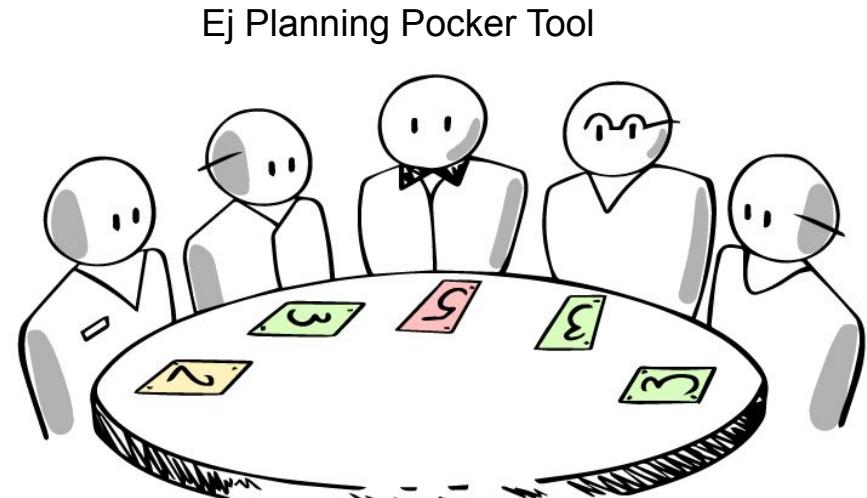
# Métodos Delphi de Predicción y Estimación



- El Método Delphi es una técnica creada por la Corporación RAND<sup>34</sup> hacia fines de la década de los 40's para la elaboración de pronósticos y predicciones sobre el impacto de la tecnología en la Guerra Fría.
- Su objetivo es lograr un consenso basado en la discusión entre expertos. Este método se basa en la elaboración de un cuestionario que ha de ser contestado por una serie de expertos. Una vez recibida la información, se vuelve a realizar otro cuestionario basado en el anterior para ser contestado nuevamente.
- El Método Delphi se basa en:
  - El anonimato de los participantes
  - La repetición y retroalimentación controlada
  - La respuesta del grupo en forma estadística

# Planning Poker

- Se basa en el método Wideband Delphi para realizar la estimación de requerimientos (o User Stories) de forma colaborativa en un Equipo.
- La técnica consiste en que cada integrante del Equipo posee en sus manos una baraja de cartas con los números correspondientes a la sucesión de Fibonacci y se siguen los siguientes pasos:
  - El responsable del negocio presenta una historia de usuario para ser estimada.
  - Todos los participantes proceden a realizar su estimación en forma secreta, sin influenciar al resto del Equipo, poniendo su carta elegida boca abajo sobre la mesa.
  - Una vez que todos los integrantes han estimado, se dan vuelta las cartas y se discuten principalmente los extremos.
  - Al finalizar la discusión se levantan las cartas y se vuelve a estimar, esta vez con mayor información que la que se tenía previamente.
  - Las rondas siguen hasta que se logra consenso en el Equipo y luego se continúa desde el punto número uno con una nueva historia de usuario.



# La Sabiduría de las Multitudes



- James Surowiecki explica: "Normalmente solemos favorecer la opinión de los expertos, pues consideramos que sólo una persona con experiencia y conocimientos suficientes es capaz de emitir juicios correctos en un área o materia en particular. Sin embargo, hay evidencias de que las decisiones tomadas colectivamente por un grupo de personas suelen ser más atinadas que las decisiones tomadas sobre la base del conocimiento de un experto".
- Dadas las circunstancias requeridas, un grupo de personas puede tomar una decisión más acertada que la mejor de las decisiones de la mayoría (si no todos) los integrantes del grupo individualmente.
- Para que esto pueda suceder:
  - Diversidad de opiniones
  - Independencia
  - Agregación

# Conclusiones sobre estimaciones Ágiles



- No tiene sentido presentar estimaciones certeras al comienzo de un proyecto ya que su probabilidad de ocurrencia es extremadamente baja por el alto nivel de incertidumbre.
- Intentar bajar dicha incertidumbre mediante el análisis puede llevarnos al “Parálisis”. Para evitar esto debemos estimar a alto nivel con un elevado grado de probabilidad, actuar rápidamente, aprender de nuestras acciones y refinar las estimaciones frecuentemente.
- La mejor estimación es la que provee el Equipo de trabajo. Esta estimación será mucho más realista que la estimación provista por un experto ajeno al Equipo.

# Incepción – Sprint 0

- La incepción es una aproximación que muchos autores utilizan para realizar todas aquellas tareas necesarias para hacer el setup de un proyecto de desarrollo.
- Esto incluye pero no se limita únicamente a:
  - configurar los entornos de desarrollo, realizar el plan de entregas, diseñar la arquitectura de la aplicación a alto nivel, configurar el repositorio de código fuente, socializar una visión común entre equipo Scrum y stakeholders,
  - crear un elevator pitch del producto, realizar un impact mapping y un visual story mapping, etc.

# Incepción – Sprint 0 - Cont.

- Aunque el término “Sprint 0” no está definido de manera específica en la Guía Scrum oficial, muchas organizaciones y equipos lo utilizan para referirse a este período previo al inicio de los sprints regulares.

¿Pero existe o no un Sprint 0 en Scrum?

- El enfoque de Scrum es comenzar con un sprint completo y funcional desde el principio, evitando un Sprint 0 en Scrum de manera formal.
- En esta perspectiva, la idea es que todos los sprints deberían ser capaces de generar incrementos potencialmente entregables desde el principio
- Lo más importante es que el equipo Scrum y la organización encuentren el enfoque que mejor se adapte a sus necesidades y contextos específicos.

Source: [scrum.org](https://scrum.org)

# Incepción – Sprint 0 - Cont.

Aquí hay algunas características y actividades comunes asociadas con el Sprint 0:

1. **Preparación del entorno:** Durante el Sprint 0, el equipo puede configurar el entorno de desarrollo, establecer herramientas y sistemas, y asegurarse de que todos los miembros tengan acceso a lo necesario para trabajar eficazmente.
2. **Definición de la arquitectura:** En este período, el equipo puede discutir y definir la arquitectura general del proyecto. Esto puede incluir decisiones sobre tecnologías, patrones de diseño y estructura del código.
3. **Recopilación de requisitos iniciales:** Se pueden llevar a cabo conversaciones con los stakeholders para identificar y priorizar las características clave del producto. Esto puede ayudar a tener una idea clara de lo que se construirá en los sprints posteriores.
4. **Identificación y preparación de historias de usuario:** Durante el Sprint 0, el equipo puede trabajar en la descomposición inicial de las características en historias de usuario más pequeñas y manejables. Esto ayuda a preparar el backlog de producto para los primeros sprints.
5. **Planificación y establecimiento de expectativas:** El equipo puede discutir y establecer las expectativas sobre cómo funcionarán los sprints, cómo se llevarán a cabo las reuniones, cuál será la duración de los sprints, entre otros aspectos.
6. **Alineación del equipo:** El Sprint 0 puede ser una oportunidad para que los miembros del equipo se conozcan mejor, comprendan sus roles y responsabilidades, y establezcan un sentido de colaboración y compromiso.
7. **Preparación para el primer sprint:** Al finalizar el Sprint 0, el equipo debería estar listo para comenzar el primer sprint oficial con un backlog preparado y definido. Esto incluye historias de usuario listas para ser trabajadas y tareas más detalladas planificadas.

source: [scrum.org](http://scrum.org)

# Impact Mapping

## Introducción

- **Impact Mapping** es una novedosa técnica de planificación y estrategia creada por [Gojko Adzic](#), un reconocido consultor en Ingeniería de Software. Adzic es, además, autor de numerosos libros sobre metodologías, especificaciones y testing, entre otras áreas. Esta técnica tuvo una gran repercusión en la comunidad Agile, con especial aplicación en la iteración 0 de los proyectos o bien antes de su inicio.
- Es una herramienta fundamental que permite, además, generar la versión inicial de un Product Backlog.

source: [Impact Mapping en Metodologías Ágiles](#)

# Impact Mapping - Cont.



## ¿Qué es y para qué sirve?

- Impact Mapping es una metodología que propone concentrar esfuerzos en lo realmente importante, en hacer las preguntas correctas para lograr determinar con precisión qué es aquello que debe conseguirse para lograr las metas deseadas
- Es una herramienta fundamental que permite, además, generar la versión inicial de un Product Backlog.
- Hacer las preguntas correctas para detectar qué es necesario conseguir y cómo, para satisfacer las metas a alcanzar.

source: [Impact Mapping en Metodologías Ágiles](#)

# Impact Mapping - Cont.

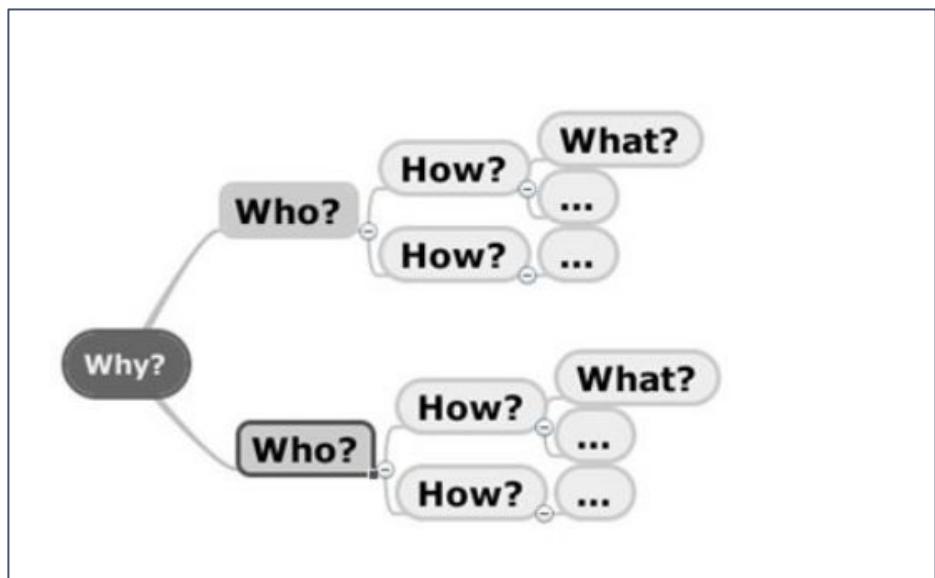
## Cuatro preguntas conducen la técnica:

1. ¿Por qué?
  - por qué se está haciendo esto
2. ¿Quién?
  - quién puede lograr el efecto deseado, quién lo puede obstruir, quién se verá beneficiado (es decir, el conjunto completo de los stakeholders)
3. ¿Cómo?
  - cómo impactará su comportamiento una vez logrado el objetivo.
4. ¿Qué?
  - Finalmente, se responde qué puede hacer la organización o empresa para apoyar y consolidar los cambios necesarios para lograr la meta deseada

source: [Impact Mapping en Metodologías Ágiles](#)

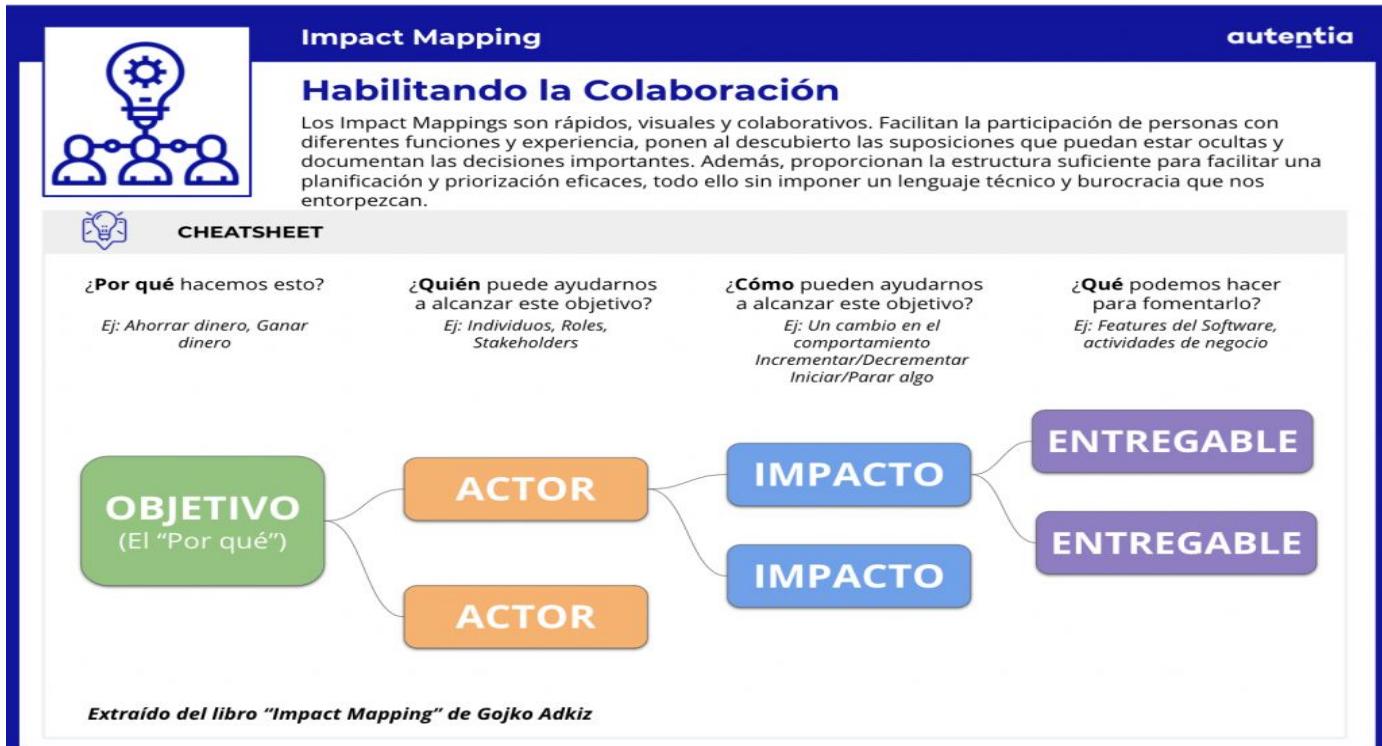
# Impact Mapping - Cont.

Toda esta información se estructura visualmente con un estilo de árbol y ramificaciones, tal como lo ilustra la Figura 1.



source: [Impact Mapping en Metodologías Ágiles](#)

# Impact Mapping - Cont.



- Se centra en la colaboración entre todas las partes interesadas.
- Visualiza y comunica claramente las suposiciones.
- Es rápido e iterativo para poder adaptar los cambios sin demora.

source: [Adictos Al Trabajo](#)

# Duración del Proyecto – Release Plan

## Release Plan:

- Se define en función de las prioridades definidas por el product owner, la duración de los sprints y la velocidad del equipo, la cantidad de sprint por release.

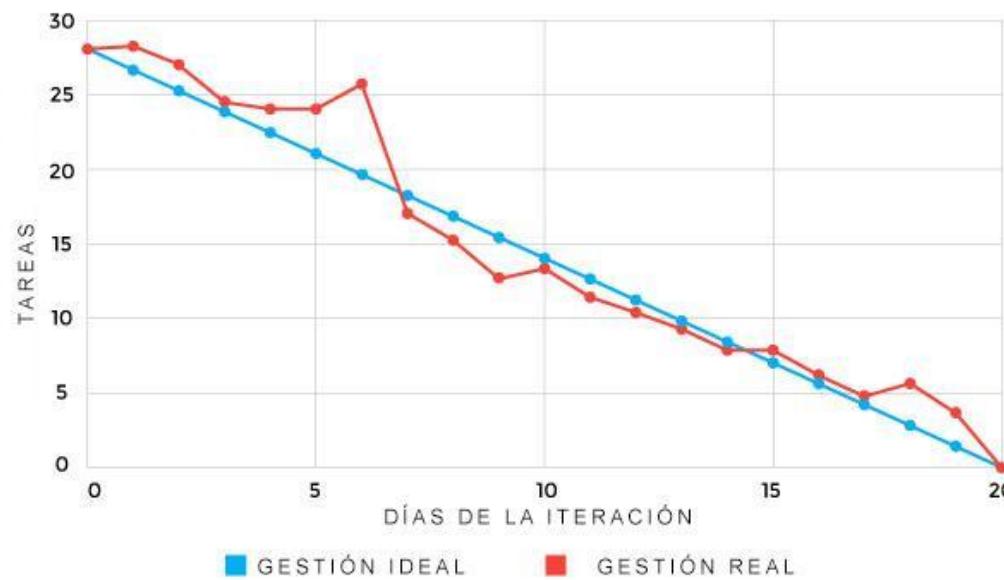
Duración Total: 18 Sprints = 36 Semanas = 9 meses

# Velocidad

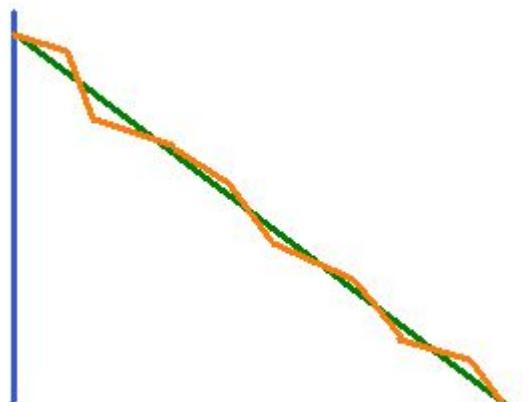
- La velocidad es el número de unidades de trabajo realizadas en un cierto intervalo.
- En un proyecto Scrum es la cantidad de puntos de la historia que el equipo de desarrollo puede completar durante un Sprint.
- La velocidad se calcula promediando el número de puntos de historia completados en los sprints anteriores. Generalmente 5 Sprints.
- En la práctica la velocidad es un valor que convierte los puntos de historia, basados en un esfuerzo relativo, en tiempo.
- Las historias no terminadas no suman.
- Es el compromiso tomado por el equipo a realizar tantos users stories durante un sprint de manera que sumen una estimación en puntos de historia igual a la velocidad del equipo.

# Burn Down Chart

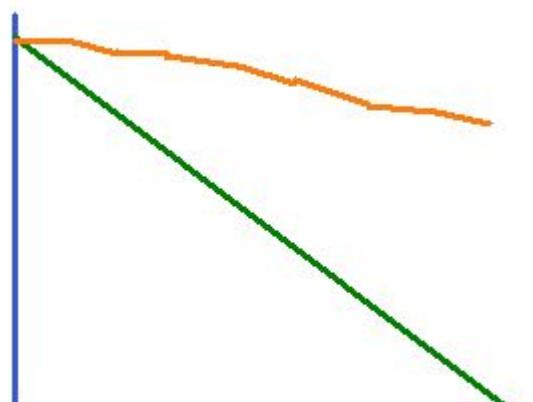
- El burndown chart es una gráfica sencilla de leer, que revela los datos de la velocidad con la que se está desarrollando un Sprint.
- Además revela los datos para conocer cuánto trabajo resta, qué tareas se completaron y el tiempo restante para terminarlas.
- Ventajas:
  - Cohesión del equipo
  - Representación visual del avance
  - Simplicidad
- "Un burndown perfecto es sospechoso."



# Diferentes Gráficas de Burn Down



Sprint Ideal



Sprint Incompleto  
Mal Estimado



Sprint con Funcionalidad Reducida  
Mal Estimado



Sprint con Funcionalidad Adicionada  
Mal Estimado

# Origen de Kanban

- Nació para aplicarse a los procesos de fabricación.
- Surgió en Toyota Production System (TPS) a finales de los años 40.
- Dentro de la implementación del modelo de producción “Just in Time”.
- Sentó las bases del “Lean Manufacturing”.
- La producción Lean es un modelo de gestión que se enfoca en minimizar las pérdidas de los sistemas de manufactura al mismo tiempo que maximiza la creación de valor para el cliente final.
- Proviene de dos palabras japonesas: kan, que significa visual, y ban, que significa tarjeta.

# ¿Qué es un tablero Kanban?

- El tablero Kanban es la herramienta para mapear y visualizar su flujo de trabajo y uno de los componentes claves del método Kanban.
- Originalmente, se utilizaba una pizarra blanca (o un tablero de corcho) que se dividía en columnas y filas.
- Cada columna visualiza una fase de su proceso y las filas representan diferentes tipos de actividades específicas (diseño, errores, deuda técnica, etc.).
- El tablero más básico de Kanban está compuesto por tres columnas: “Por hacer”, “En proceso” y “Hecho”.
- Demuestra dónde están los cuellos de botella en el proceso y qué es lo que impide que el flujo de trabajo sea continuo e ininterrumpido.

# Método Kanban

- Se demostró que Kanban era conveniente no solo para la industria automotriz, sino también para cualquier otro tipo de industria. Así es como nació el método Kanban.
- David J. Anderson (reconocido como el líder de pensamiento de la adopción del Lean/Kanban para el trabajo de conocimiento) formuló el método Kanban como una aproximación al proceso evolutivo e incremental y al cambio de sistemas para las organizaciones de trabajo.
- El método está enfocado en llevar a cabo las tareas pendientes y los principios más importantes pueden ser divididos en cuatro principios básicos y seis prácticas

# Kanban: Los 4 Principios

- **Empezar con lo que hace ahora:**

- Kanban no requiere configuración y puede ser aplicado sobre flujos reales de trabajo o procesos activos para identificar los problemas. Por eso es fácil implementar Kanban en cualquier tipo de organización, ya que no es necesario realizar cambios drásticos.

- **Comprometerse a buscar e implementar cambios incrementales y evolutivos**

- El método Kanban está diseñado para implementarse con una mínima resistencia, por lo que trata de pequeños y continuos cambios incrementales y evolutivos del proceso actual. Los cambios radicales no son considerados.

# Kanban: Los 4 Principios

- **Respetar los procesos, las responsabilidades y los cargos actuales**

- Kanban reconoce que los procesos en curso, los roles, las responsabilidades y los cargos existentes pueden tener valor y vale la pena conservarlos
- El método Kanban no prohíbe el cambio, pero tampoco lo prescribe.
- Alienta el cambio incremental, ya que no provoca tanto miedo como para frenar el progreso.

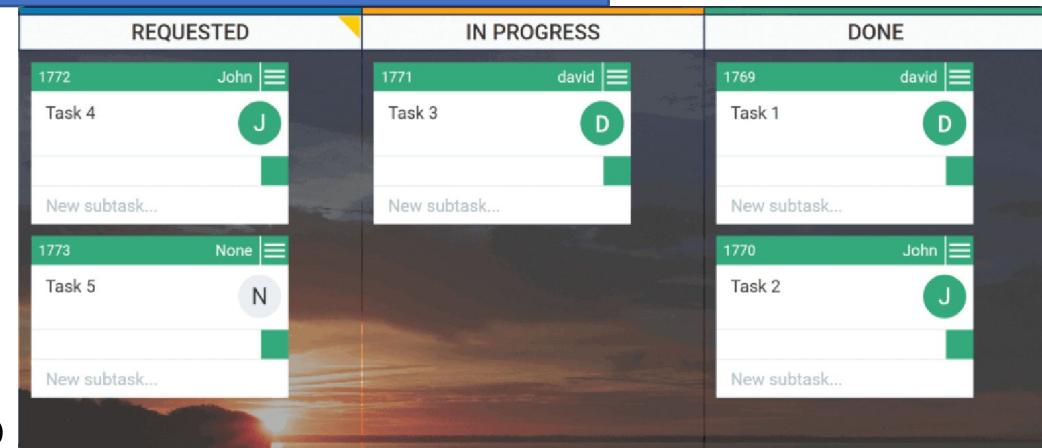
- **Animar el liderazgo en todos los niveles**

- Este es el principio más novedoso de Kanban.
- Algunos de los mejores liderazgos surgen de actos del día a día de gente que está al frente de sus equipos.
- Es importante que todos fomenten una mentalidad de mejora continua (Kaizen).

# Las 6 prácticas de Kanban

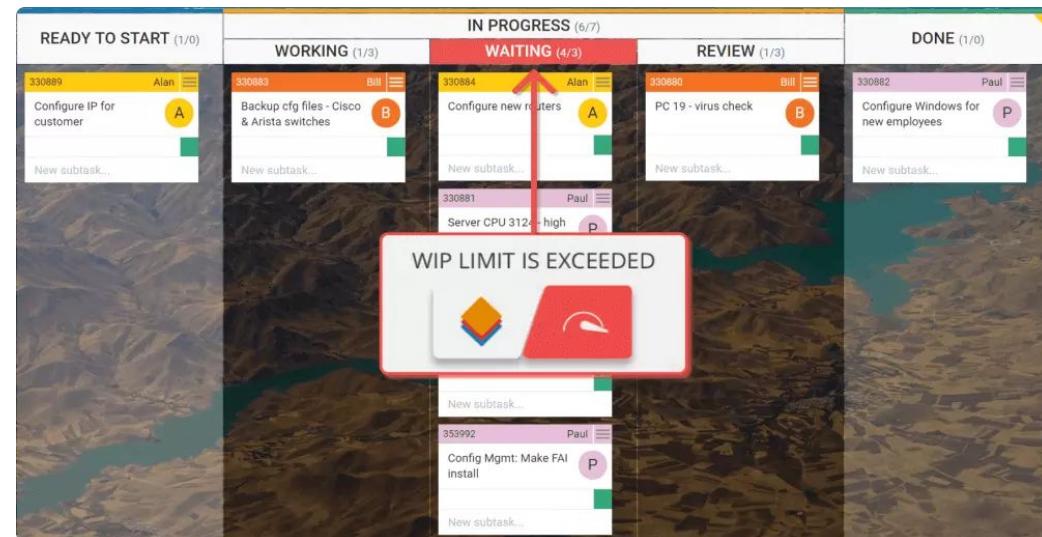
- Visualizar el flujo de trabajo

- Lo mas importante es comprender el proceso
- Solo después de entender cómo funciona actualmente el flujo de trabajo, puede aspirar a mejorarlo haciendo los ajustes necesarios.
- Para visualizar su proceso en Kanban, necesitará un tablero con tarjetas y columnas.
- Cada columna del tablero representa un paso en su flujo de trabajo
- Cada tarjeta Kanban representa un elemento de trabajo.
- De esta forma, puede fácilmente seguir el progreso y detectar los cuellos de botella.



# Las 6 prácticas de Kanban

- Eliminar las interrupciones
  - Establecer los límites del trabajo en proceso (los límites WIP).
  - La abreviatura WIP significa “Work In Progress”
  - Los límites restringen la cantidad máxima de elementos de trabajo en las diferentes etapas (columnas del tablero Kanban) del flujo de trabajo.
  - Ayuda al equipo a enfocarse solo en las tareas actuales y así le permite terminar más rápido con los elementos de trabajo individuales.
  - Localizar los cuellos de botella en los procesos de trabajo antes de que éstos se conviertan en bloqueos.
  - Asegura que una tarjeta se “arrastra” al siguiente paso sólo cuando hay capacidad disponible.



# Las 6 prácticas de Kanban

- Gestionar el flujo
  - El objetivo es crear un flujo continuo e ininterrumpido.
  - Lo que interesa es la velocidad y la continuidad del movimiento.
  - Esto significaría que nuestro sistema está creando valor rápidamente
- Hacer las políticas explícitas (Fomentar la visibilidad)
  - No puede mejorar algo que no se entiende.
  - El proceso debe estar bien definido, publicado y promovido

# Las 6 prácticas de Kanban

- Circuitos de retroalimentación

- La filosofía Lean admite que las reuniones regulares son necesarias para la transferencia de conocimiento (circuitos de retroalimentación).
- Tales son las reuniones diarias de pie para sincronizar el equipo.
- Se llevan a cabo frente al tablero Kanban y cada miembro comparte con los demás lo que él o ella hizo el día anterior y qué va a hacer el día de hoy.
- También existen las reuniones para la revisión de entrega de servicios, la revisión de operaciones y la revisión de riesgos.
- Su frecuencia depende de muchos factores, pero la idea es que sean regulares, a una hora estrictamente fija, directas al grano y nunca innecesariamente largas.
- La duración promedio ideal de una reunión de pie debe ser entre 10 y 15 minutos, y las demás reuniones pueden durar hasta una hora, en función del tamaño del equipo y los temas.

# Las 6 prácticas de Kanban

- Mejorar colaborando
  - Visión compartida para un futuro mejor y la comprensión colectiva de los problemas que deben superarse.
  - Los equipos que tienen un entendimiento compartido de las teorías sobre el trabajo, el flujo de trabajo, el proceso y el riesgo, tienen más probabilidades de crear una comprensión compartida de un problema y sugerir acciones de mejora que pueden acordarse por consenso.

# Riesgos: definición

- La mejor manera de evitar riesgos en los proyectos es enfrentandolos con anticipación. La planificación proactiva de este tipo de casos puede ser útil para encauzar el proyecto.
- La gestión de riesgos no es reactiva, es proactiva
- Los riesgos son eventos o condiciones inciertas que, si se producen, tienen un efecto positivo o negativo sobre al menos un objetivo del proyecto, como tiempo, coste, alcance o incluso la calidad.

# Gestión del Riesgo

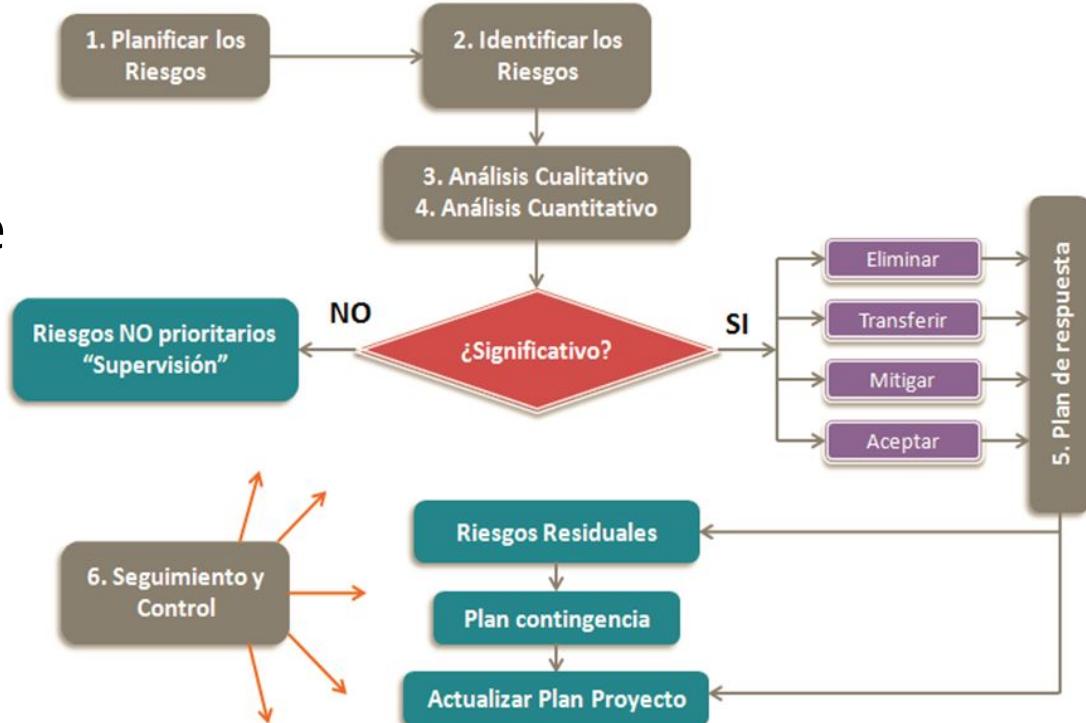
¿Qué significa gestionar el riesgo de un proyecto?

- Desarrollar:
  - Estrategias de Administración de Riesgos
  - Acciones de contingencia.
- Monitorear constantemente factores de riesgo del proyecto
- Recordar:

**Si RIESGO entonces IMPACTO**

# Pasos a seguir

- **Identificación del riesgo.**
- **Evaluación:** Analizar cada riesgo y establecer probabilidad de ocurrencia y el daño que causará si ocurre.
- **Ordenamiento:** Realizar una clasificación de los riesgos según su probabilidad de ocurrencia y el impacto que pudiesen causar.
- **Planificación de las respuestas:** Desarrollar un plan de para determinar que acciones tomar frente a aquellos riesgos con gran probabilidad de que ocurran.
- **Monitoreo y control**



# Estrategias de identificación

- Estrategias reactivas
  - Método:
    - Evaluar las consecuencias del riesgo cuando éste ya se ha producido (ya no es un riesgo)
    - Actuar en consecuencia
  - Consecuencias de una estrategia reactiva
    - Apagado de incendios
    - Se pone el proyecto en peligro
- Estrategias proactivas
  - Método
    - Evaluación previa y sistemática de riesgos
    - Evaluación de consecuencias
    - Plan de evitación y minimización de consecuencias
    - Plan de contingencia
  - Consecuencias
    - Evasión del riesgo
    - Menor tiempo de reacción

# Evaluación

- Analizar las probabilidades de ocurrencia:
  - 1 = poco probable, 5 = muy probable
- Analizar el impacto de cada riesgo y ponderar:
  - 1 = muy bajo impacto, 5 = muy alto impacto
- Como salida tendremos cada riesgo ponderado en sus dos dimensiones:
  - Probabilidad
  - Impacto

# Ordenamiento

- Multiplicar: Probabilidad \* Impacto
- Ordenar por prioridad
- Definir que riesgos necesitan un plan de respuesta

# Planificación de Respuestas

- Se debe elegir la estrategia de como será tratado cada riesgo:
  - Evitar / Explorar
  - Transferir / Compartir
  - Mitigar / Mejorar
  - Aceptar / Aceptar
- Riesgo Negativo – AMENAZA
  - Evitar: Eliminar la causa
  - Mitigar: Reducir la probabilidad o el impacto
  - Transferir: Poner en otra parte la responsabilidad (Seguro, Tercerizar, etc)
  - Aceptar: Aceptar las consecuencias
    - Activo: Plan de contingencia
    - Pasivo: Aceptar que ocurra.

# Planificación de Respuestas

- Riesgo Positivo – Oportunidad

- Explorar: Eliminar la incertidumbre. Lograr que la oportunidad se concrete
- Compartir: Trabajar con un tercero que esté mas capacitado para capturar la oportunidad.
- Mejorar: Modificar una oportunidad. Aumentando Probabilidad o Impacto. Fortalecer la causa de la oportunidad
- Aceptar: Tomar ventaja de ella en la medida que ocurre.

# Causas Comunes

- Cambios en los requerimientos
- Errores de Diseño u omisiones relacionados al alcance
- Mala definición de roles y responsabilidades
- Estimaciones incorrectas
- Equipo no entrenado lo suficiente
- Usuarios finales no entrenados lo suficiente.



Microsoft Excel  
Worksheet