INGENIERÍA Y CALIDAD DE SOFTWARE

INTRODUCCIÓN

Software → set de programas y la documentación que acompaña.

Existen 3 tipos básicos de software:

- System software
- Utilitarios
- Software de aplicación

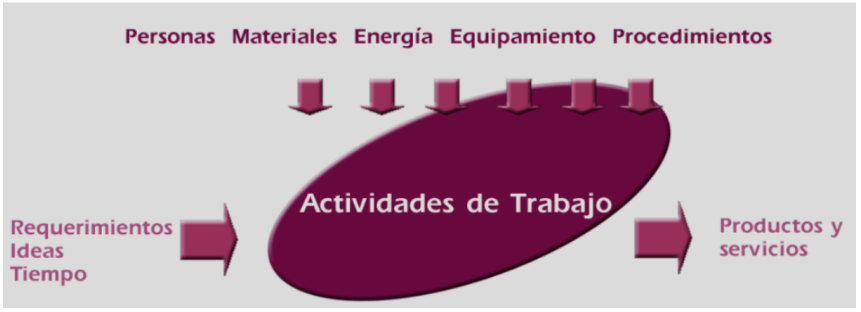
El software no se produce en masa, casi ningún producto de software es igual a otro. No todas las fallas son errores, el software no se gasta, ni está gobernado por leyes de la física y es menos predecible. Todas estas son diferencias entre un software y manufactura.

Problemas con el desarrollo de software

- La versión final del producto no satisface las necesidades del cliente
- No es fácil extenderlo y/o adaptarlo. Agregar más funcional en otra versión es casi una misión imposible.
- Mala documentación
- Mala calidad
- más tiempos y costos que los presupuestados.

INGENIERÍA DE SOFTWARE - dividida en 3 ramas

PROCESO DE SOFTWARE



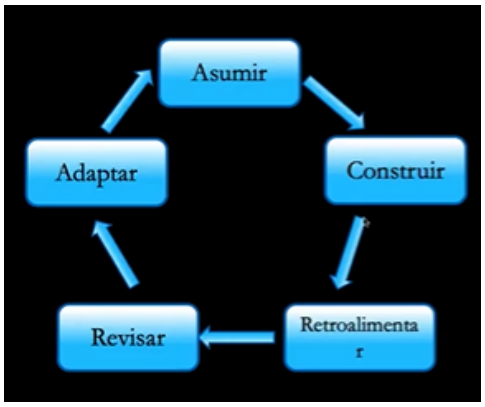
Conjunto estructurado de actividades para desarrollar un sistema de software (en base a un conjunto de entradas nos permite producir alguna salida)

Las actividades dependen de la org (el PUD es un ejemplo)

Proceso: La secuencia de pasos ejecutados para un propósito dado (IEEE)
Proceso de Software: Un conjunto de actividades, métodos, prácticas, y transformaciones que la gente usa desarrollar o mantener software y sus productos asociados (Sw-CMM)

Hay varios inputs, no solo los requerimientos, sino también los recursos humanos, materiales, librerías, etc.

Proceso empírico

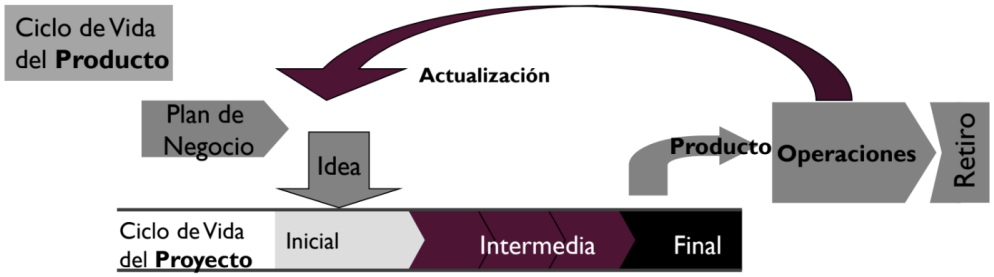


Se define a medida que vamos realizando el proceso. Centra el foco en la experiencia, tanto del usuario como de los programadores, líderes del proyecto, o sea del equipo de trabajo… (experiencia del negocio)
Se trata de aprender de la experiencia e ir mejorando.
La administración y control es mediante inspecciones frecuentes y adaptación del proceso
se usa para procesos de vida complejos y creativos con variables cambiantes
Se trabaja con ciclos de vida iterativos. Cuando se repite el proceso, se pueden llegar a obtener resultados diferentes.

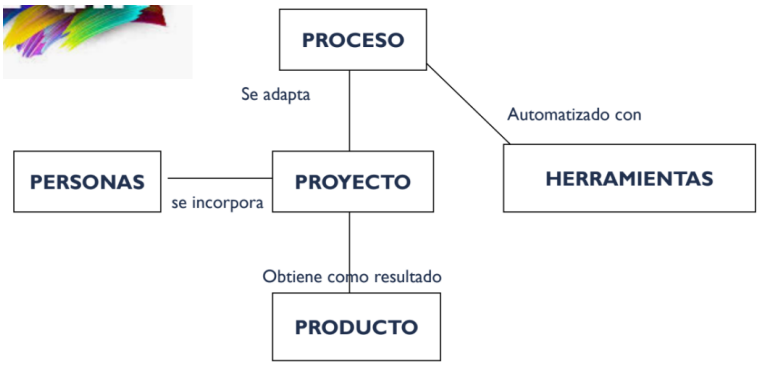
Proceso definido (gestión tradicional de proyectos)
Inspirado en las líneas de producción industrial, repetimos el mismo proceso una y otra vez y siempre obtenemos el mismo resultado. (ante la misma entrada, la misma salida). Hay una secuencia de actividades que se repiten. Ciclos de vida en cascada. Es un proceso predecible. Ejemplo: PUD.

CICLO DE VIDA → La serie de pasos a través de los cuales el producto o proyecto progresa.
Tanto los PRODUCTOS como los PROCESOS tienen ciclos de vida. EL ciclo de vida nos indica cuales son las etapas(fases) y cual es el orden.

RELACIÓN: CICLO DE VIDA DEL PROYECTO Y DEL PRODUCTO



Ciclo de vida de un proyecto: instanciación del proceso, q voy a hacer, como , pasos
cuando el proyecto termina inicia el Ciclo de vida del **producto**, hay q usarlo, probarlo, mantenerlo, actualizarlo hasta q haya q retirarlo pq queda obsoleto o no es más rentable para el negocio



un PROYECTO tiene como resultado un PRODUCTO
en un PROYECTO participan personas

el PROCESO se adapta al PROYECTO, está automatizado con herramientas

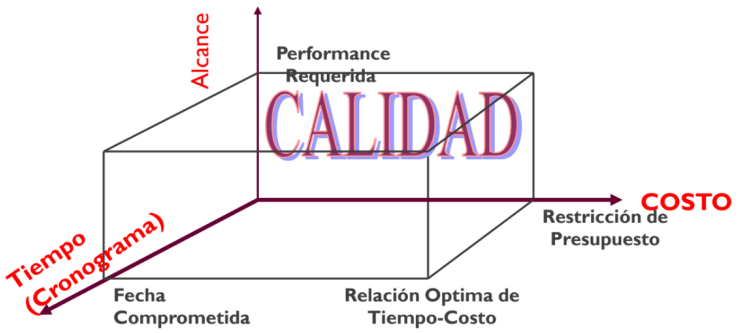
PROYECTO (características):

- Duración limitada: tiene inicio y fin(cuando se cumplen los objetivos)
- Es único, aunque puedan tener características similares.
- Orientación a objetivos: está dirigido a cumplir con un objetivo no ambiguo, y alcanzable. Dirigido a obtener resultados. Los objetivos guían al proyecto.
- Las tareas que se lleva a cabo están interrelacionadas, basadas en esfuerzo y recursos.

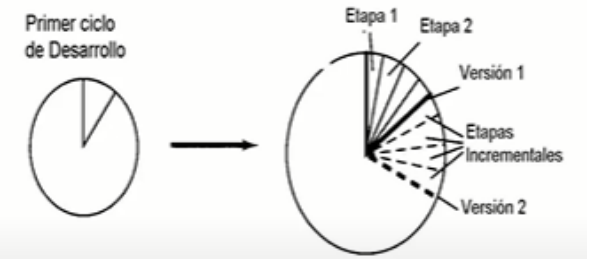
Administración de proyectos
es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para "....tener el trabajo hecho...." en tiempo, con el presupuesto acordado y habiendo satisfecho las especificaciones o requerimientos.

Administrar un proyecto incluye:
- Identificar los requerimientos
- Establecer objetivos claros y alcanzables
- Adaptar las especificaciones, planes y el enfoque a los diferentes intereses de los *involucrados (stakeholders).*

"LA RESTRICCIÓN TRIPLE" (THE TRIPLE CONSTRAINT)
Objetivos de proyecto/ Alcance: que está el proyecto tratando de alcanzar? el resultado
Tiempo: cuánto tiempo debería llevar completar el proyecto
Costos: ¿cuánto debería costar? (recursos y $)



El balance de estos tres factores afecta directamente la ***calidad*** del proyecto. La calidad nunca se negocia.
Por ejemplo, si yo quiero terminar mi proyecto antes necesito variar o el alcance o el costo o ambas.



El desarrollo de software → cada versión del producto de software es desarrollada incrementalmente en una serie de pasos.

LÍDER DEL PROYECTO (PM): gestiona al equipo de proyecto, constantemente interactúa con ellos. además interactúa con el cliente, subcontratistas, las organizaciones regulatorias, niveles altos de administración como gerentes funcionales.
El **equipo del proyecto** es el conjunto de personas comprometidas a alcanzar un conjunto de objetivos de los cuales se sienten responsables mutuamente.
Características :
• Diversos conocimientos y habilidades
• Posibilidad de trabajar juntos efectivamente / desarrollar sinergia
• Usualmente es un grupo pequeño
• Tienen sentido de responsabilidad como una unidad

PLAN DE PROYECTO → es como una hoja de ruta, es algo vivo, que puede ir cambiando a medida que avanzó en el proyecto. Es un camino" a seguir para terminar el proyecto y cumplir los objetivos . Ayuda a cumplir de la mejor manera con la triple restricción y a tomar decisiones por si surgen imprevistos
- Documenta: QUÉ hacemos, CUÁNDO lo hacemos, CÓMO lo hacemos y QUIENES lo van a hacer

La planificación de proyectos implica: Definición del alcance de proyecto, definición de proceso y ciclo de vida, estimación, gestión de riesgos, asignación de recursos, programación de proyectos, definición de controles y definición de métricas.

Definición del Alcance del Proyecto → el ALCANCE del producto no es el mismo que el del proyecto. Hay que definir los requerimientos que van a ser el alcance del producto. No necesariamente tengo todos al inicio del proyecto, pueden ir surgiendo a medida que avanza o otros puede irse modificando

Alcance del PRODUCTO: Son todas las características que pueden incluirse en un producto o servicio.
Se mide contra la Especificación de Requerimientos.
Alcance del PROYECTO: Es todo el trabajo y solo el trabajo que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas.
El cumplimiento se mide contra el Plan de Proyecto (o Plan de Desarrollo de Software).

más sobre alcance de producto y alcance de proyecto…
El alcance del producto se mide en términos de requerimientos. El alcance del proyecto se mide en términos del trabajo que hay que realizar en el contexto de ese proyecto para obtener como resultado el producto que espero.
Cuando hablamos del alcance del producto, su vida sobrevive al proyecto, porque mientras el producto funcione mantengo los requerimientos. En cambio, cuando el proyecto termina no lo uso nunca más, lo que me interesa es el producto.

Definición de Proceso y Ciclo de Vida → Un ciclo de vida de un proyecto software es una representación de un proceso. Grafica una descripción del proceso desde una perspectiva particular.
Se especifican las fases de proceso, y el orden en que se llevan a cabo.
Hay *tres tipos básicos* de Ciclos de Vida para un proyecto de desarrollo de software
Clasificación de los ciclos de vida:

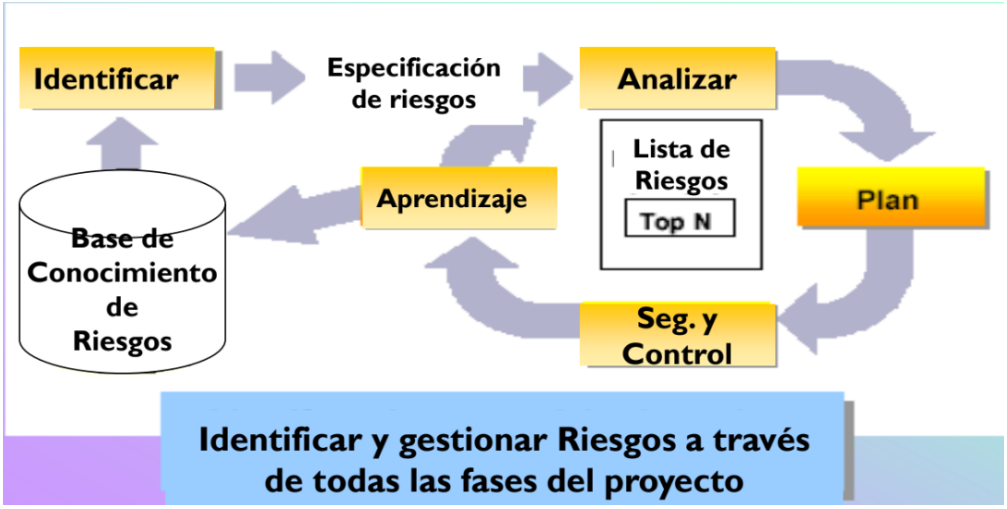
- secuencial: se basa en una etapa después de otrora y no tienen retorno generalmente.
- iterativo/incremental: como la mayoría de los procesos empíricos, en cada iteración se capitaliza lo que aprendemos, mejoramos la experiencia y seguimos una iteración más para incrementar el producto. Acá se repite y se avanza, repite y avanza.
- Recursivo: para proyectos que tienen altos riesgos, por ejemplo ciclo de vida en espiral. se basa en ir metiéndote un poquito más adentro en cada vuelta. Tomas una característica del producto, un riesgo en particular, haces una iteración para solucionarlo y luego se indaga más profundo para mejorar otra cosa.

— *Capítulo 7 de Desarrollos de proyectos informáticos (Rapid Development) de Mcconell* —

Relación entre producto, proceso y ciclo de vida → el proceso es una implementación del ciclo de vida. El ciclo de vida es una abstracción, es lo que nos guía en cuales son las etapas y el orden, y el proceso nos dice cómo hacerlo. Todo esto para obtener el resultado final: el producto.

Estimación → cuando creamos software estimamos : Tamaño, esfuerzo, tiempo , costos, recursos críticos. Es una base que sirve de input.

Gestión de Riesgos → Problema esperando para suceder que podría comprometer el éxito del proyecto, tiene asociada una probabilidad de ocurrencia la cual puede variar a lo largo de la vida del proyecto.



Asignación de Recursos
Programación de Proyectos

Definición de Controles
Definición de Métricas (se ve bien en otra unidad) se definen para medir en términos concretos saber como vamos con el proyecto
El dominio de las métricas del software se divide en:
-Métricas de proceso:
-Métricas de proyecto: me permiten ver como voy con el desarrollo del mi proyecto
-Métricas de producto.
Las métricas del PROYECTO se consolidan para crear métricas de PROCESO que sean públicas para toda la organización del software.
Métricas básicas para un proyecto de software: Tamaño del producto - Esfuerzo (es una métrica de proyecto) - Tiempo (es métrica de proyecto) - Defectos

Tres factores para el **éxito** de un proyecto
•Monitoreo & Feedback
•Tener una misión/objetivo claro
•Comunicación

Causas de fracasos en un proyecto
Fallas al definir el problema
Planificar basado en datos insuficientes
La planificación la hizo el grupo de planificaciones
No hay seguimiento del plan de proyecto
Plan de proyecto pobre en detalles
Planificación de recursos inadecuada
Las estimaciones se basaron en “supuestos” sin consultar datos históricos
Nadie estaba a cargo

CLASE TEÓRICA REQUERIMIENTOS ÁGILES Y US

Los 12 principios del Manifiesto Ágil

- Satisfacer al cliente con entregas frecuentes y tempranas
- Cambios de requerimientos son bienvenidos
- releases frecuentes (de 2 a 4 semanas)
- Técnicos y no técnicos juntos
- individuos motivados
- medio de comunicación: cara a cara
- métrica de progreso: software funcionando
- ritmo de desarrollo sostenible
- atención continua a la excelencia técnica
- simplicidad: maximización del trabajo no hecho
- arquitecturas, diseños y requerimientos emergentes
- a intervalos regulares el equipo evalúa su desempeño.

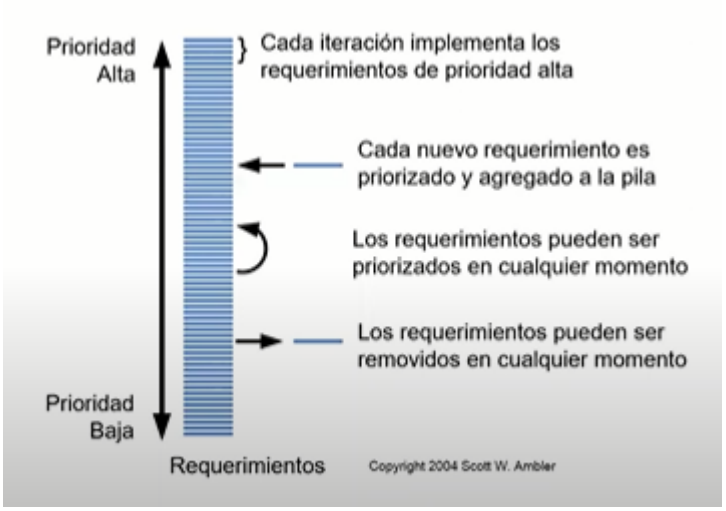
Requerimientos en Agile

Uso del valor: cuando hablamos de Requerimientos Agile hay algo que nos delimita al definir los requerimientos, tiene que ver con aquello que le da valor al producto que estamos construyendo.
Usar el “valor” para construir el producto correcto.
Usar historias y modelos para mostrar que construir
Determinar que es “sólo lo suficiente”, no trabajar de más.

El costo del tradicional BRUF

Los productos “exitosos” tienen un desperdicio significante, hay funcionalidades que no se utilizan nunca.

Gestión Ágil de Requerimientos de software → Los requisitos cambiantes son una ventaja competitiva si puede actuar sobre ellos



Los que están en el tope de la lista, son aquellos que tengo que trabajar primero con más detalle. Lo que queda más abajo en la pila va a poder ir cambiando en el tiempo hasta que tenga la suficiente prioridad y ahí voy a avanzar en su detalle y más adelante en su implementación.

JUST IN TIME: analizo cuando lo necesito, no antes ni después(que no se haga tarde). analizo con detalle un requerimiento cuando tenga el suficiente valor.

Otro principio Ágil: La forma de comunicación CARA-CARA

- El cara-cara permite que fluya información vocal, subvocal, gestual, con retroalimentación rápida.

Tradicional vs Ágil

	Tradicional	Ágil
Prioridad	Cumplir el plan.	Entregar Valor.
Enfoque	Ejecución ¿Cómo?	Estrategia (¿Porqué? ¿Para qué?).
Definición	Detallados y cerrados. Descubrimientos al inicio.	Esbozados y evolutivos – Descubrimi progresivo.
Participación	Sponsor, stakeholder de mayor poder e interés.	Colaborativo con stakeholders de ma (clientes, usuarios finales).
Equipo	Analista de Negocios, Project Manager y Áreas de Proceso.	Equipo multidisciplinario.
Herramientas	Entrevistas, observación y formularios.	Principalmente prototipado. Técnicas de facilitación para descubrir.
Documentación	Alto nivel de detalle – Matriz de Rastreabilidad para los Requerimientos	Historias de Usuario Mapeo de Historias (Story Mapping)
Productos	Definidos en alcance	Identificados progresivamente
Procesos	Estables, adversos al cambio	Incertidumbre, abierto al cambio

En un proyecto tradicional, dejo fijo el alcance (requisitos), a partir de ese alcance que quiero construir defino los recursos y el tiempo. Esto sucede con la **Triple Restricción.**
En la gestión ágil, las variables son las mismas, pero primero me guío por el valor. El alcance no queda fijo, dejo fijo los recursos y el tiempo. Tengo estos recursos y este tiempo, con eso ¿que le puedo entregar al cliente?

Tipos de requerimientos

- requerimientos de negocio
- requerimientos de usuario
- Requerimiento funcional: los que se modelan con los CU
- requerimiento no funcional: que no quedan plasmada en un cCU directamente, tiene que ver con características de software y no con la usabilidad que se le entrega al usuario
- requerimientos de implementación

En resumen, ¿que necesito saber de los requerimientos?

- necesito entender las necesidades del negocio, es lo que le da valor al negocio
- En los req ágiles busco entender qué le da valor a mi producto, voy priorizando los requerimientos (lo de mas arriba tiene mas valor), trabajo con un equipo competente y todo esto para hacer entregas frecuentes de software funcionando a los clientes.

TIPS

- Los cambios son la única constante. Los cambios son parte.
- Stakeholders(todos los involucrados del proyecto): no son todos los que están, roles, personas organizaciones que las percibamos como parte del equipo o no. Por ejemplo: si nos metemos con algo del ambiente un stakeholder va a ser una org del ambiente que no es ni cliente ni usuario nada
- Siempre se simple eso de que: “el usuario dice lo que quiere cuando recibe lo que pidió”
- No hay técnicas ni herramientas que sirvan para todos los casos.
- Lo importante no es entregar una salida, un requerimiento, lo importante es entregar un resultado, una solución de valor.

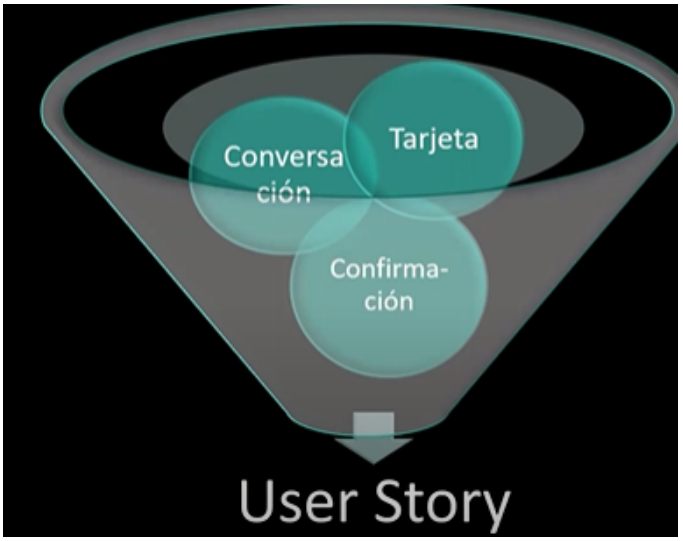
Principios Ágiles (del manifiesto ágil) relacionados a los Requerimientos Ágiles

1. La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a un mes)
2. recibir cambios de requerimientos, aun en etapas finales
3. técnicos y no técnicos trabajando juntos todo el proyecto
4. el medio de comunicación por excelencia es cara a cara
5. las mejores arquitecturas, diseños y requerimientos emergen de equipos autoorganizados

USER STORIES → herramienta que se usa como si contáramos una historia, lo importante no es lo que escribo sino todo lo que se habla sobre la historia.

La parte más difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados… Ninguna otra parte de trabajo afecta tanto al sistema resultante si se hace incorrectamente.

¿Qué son las user stories? es una manera de trabajar con req ágiles y que tiene que ver con contar historias.
¿Cuáles son sus partes? (en el frente de la tarjeta nos encontramos con la descripción y los criterios de aceptación)



LAS 3 C, la conversación es la parte más importante.

- **Tarjeta**: ahí escribimos algo que nos indique cual es la historia de usuario, de que se trata y el valor de negocio. Es donde se describe la user story, es un soporte físico.

Forma de expresar las historias de usuario:
Como<nombre del rol> yo puedo<actividad> de forma tal que <valor de negocio que recibo>
. nombre del rol: representa quien está realizando la acción o quien recibe el valor de la actividad
. actividad: representa la acción que realizará el sistema
. valor de negocio que recibo: comunica porque es necesaria la actividad.

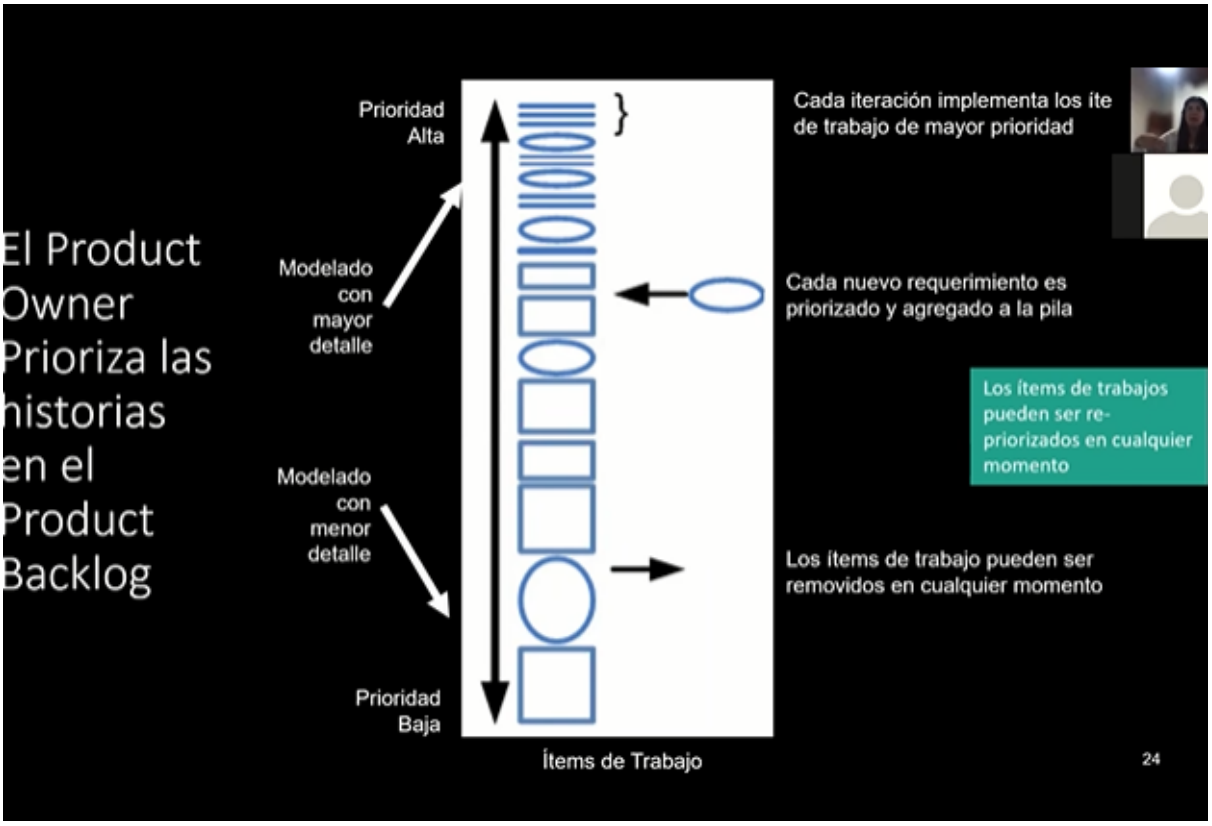
Como conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

- **Conversación**: la parte más importante. Es el momento en el cual se dialoga con el usuario (con el que ejerce el rol de product owner??) para captar las necesidades del problema, aca se obtienen los detalles para que el equipo pueda trabajar esa historia.
- **Confirmación:** condiciones que se tienen que dar para satisfacer la user story (para que esta finalice, para que esté hecha). LA CONFIRMACIÓN ESTÁ EN LA PARTE DE ATRÁS DE LA TARJETA (PRUEBAS DE ACEPTACIÓN)

Con estas 3 C ya es suficiente para comenzar a trabajar. La C más importante para el equipo es la conversación, sin esta no habrá comprensión.

Las US son multipropósito (para que nos sirven)
las historias son:

- una necesidad del usuario
- una descripción del producto
- un ítem de planificación, porque priorizamos las historias y cada historia en sí misma es un item…
- Token para una conversación
- mecanismo para diferir una conversación.



CARACTERÍSTICAS:

- Las USER STORIES capturan las necesidades e ideas de los usuarios pero no llegan a ser especificaciones detalladas de los requerimientos (como los cu).
- Son expresiones de intención, (“es necesario que haga algo mo esto…”)
- No están detallados al principio del proyecto, elaborados evitando especificaciones anticipadas, demoras en el desarrollo, inventario de requerimientos y una definición limitada de la solución.
- Necesita un poco o nulo mantenimiento y puede descartarse después de la implementación
- Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

Criterios de aceptación de Historias de Usuario → sirven para tratar de expresar reglas de negocio, consideraciones que tienen que cumplirse en esa US

- Definen límites para una User Story(US)
- Ayudan a que los PO respondan lo que necesitan para que la US provea valor (requerimientos funcionales mínimos)
- Ayudan a que el equipo tenga una visión compartida de la US
- Ayudan a desarrolladores y testers a derivar las pruebas
- Ayudan a los desarrolladores a saber cuando para de agregar funcionalidad en una US

Continuando con el ejemplo de conductor de uber:
criterios de aceptación:

- la altura de la calle es un número
- la búsqueda no puede demorar más de 30 segundos

¿Cuáles son los criterios de aceptación buenos?

- Definen una intención, no una solución. Ej: el usuario debe elegir al menos una cuenta para operar
- Son independientes de la implementación
- relativamente de alto nivel, no es necesario que se escriba cada detalle.

¿Y los detalles?¿Dónde van? NO VAN DETALLES EN LOS CRITERIOS DE ACEPTACIÓN
detalles como:

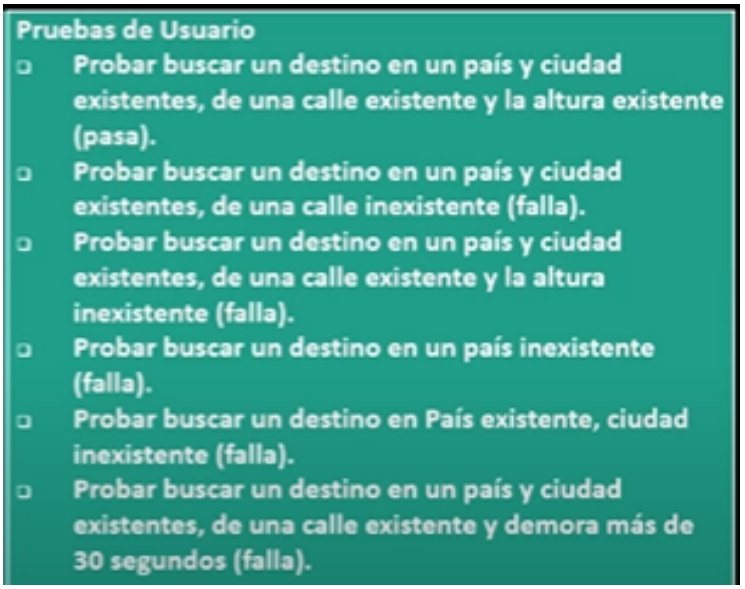
- El encabezado de la columna se nombra “saldo”
- El formato del saldo es 999.999.999,99
- debería usarse una lista desplegable en lugar de un check box

Estos detalles que son el resultado de las conversaciones con el PO y el equipo puede capturarlos en dos lugares:

- documentación interna de los equipos
- Pruebas de aceptación automatizadas.

Pruebas de Aceptación de US → expresan detalles resultantes de la conversación. Complementan la US. Es un proceso de 2 pasos:
1. identificarlas al dorso de la US
2. Diseñar las pruebas completas

Seguimos con el ejemplo del conductor - Pruebas de usuario



Se usan los verbos en infinitivo como “probar”

Definición de listo - Definition of Ready

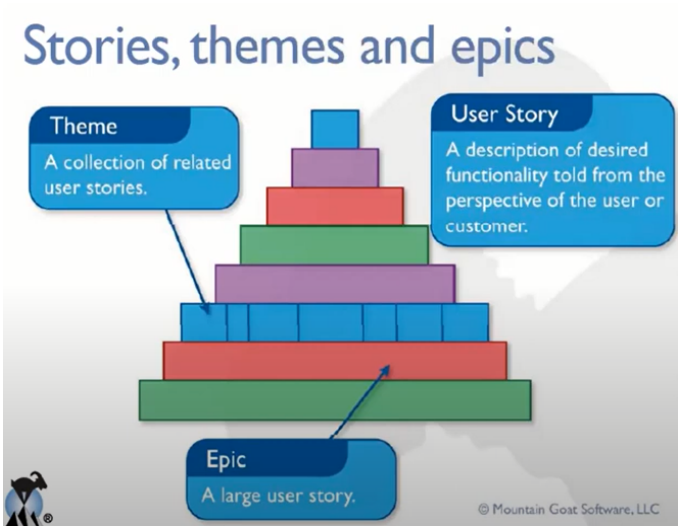
Tiene que ver con un criterio (para decir esta US la podemos tomar en esta iteración) que acordamos como equipo sobre la US. Es cuando la US está en condiciones de ser implementada dentro de un sprint, puede ser incluida en una release (ya que cumple con ciertos criterios está lista para ser desarrollada). Algo que nos ayuda a definir este criterio es INVEST MODEL

***INVEST Model:** sirve para saber si podemos incluir una US en una iteración que estamos trabajando

- *Independent:* calendarizables e implementables en cualquier orden. Que la US la puedo ejecutar en cualquier orden, la puedo poner en cualquier momento y no tiene dependencias con otras. Si es independiente está bien escrita.
- *Negotiable*: el “qué” no el “cómo”
- *Valuable*: debe tener valor para el cliente
- *Estimable*: que podemos dimensionar la complejidad, el esfuerzo que tiene para saber cómo encararla. Para ayudar al cliente a armar un ranking basado en costos. Para saber si se puede implementar en un sprint, si no se puede hay que dividirla.
- *Small*: deben ser “consumidas” en una iteración. SI no entra en una iteración la US es mas grande
- *Testable*: demostrar que se desarrolló, que sea tangible. El resultado de la US tiene que ser demostrable. Demostrar que fueron implementadas.

Definición de Hecho - definition of Done

Es propia del equipo, se valida con un checklist. Indica si la historia está decentemente terminada para presentársela al PO. Si cumple con el checklist acordado por el equipo se cumple el criterio de done y se puede presentar al PO.

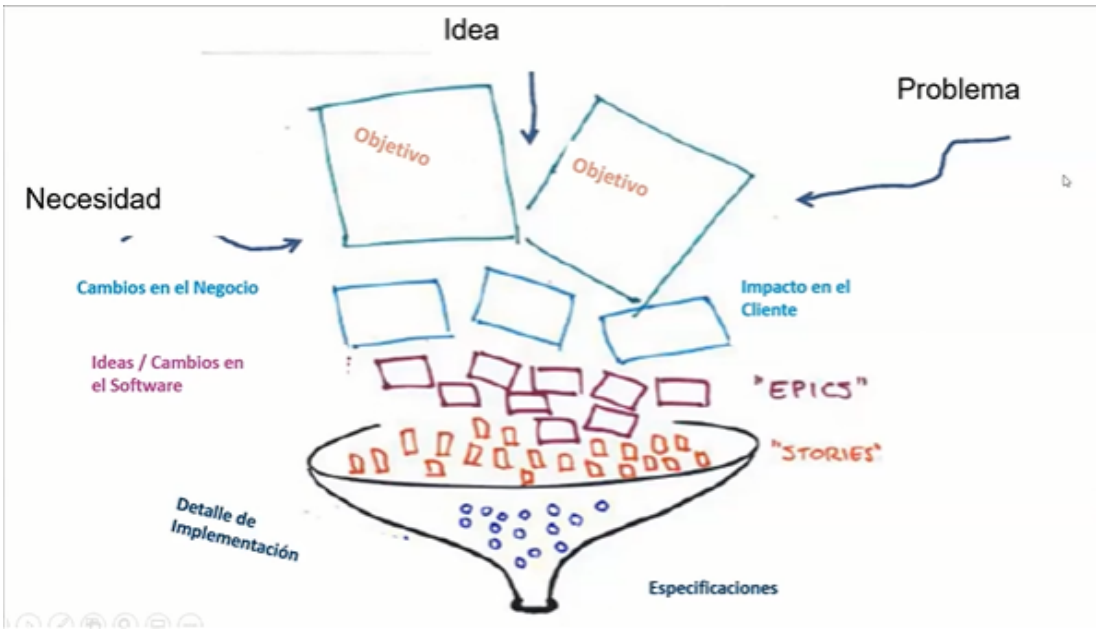


Diferentes niveles de abstracción

Cuando empiezo a trabajar me doy cuenta que me encuentro con que tengo muchas Épicas
Epicas → Historia de Usuario GRANDE, sabemos que es grande porque no se cumple que es INVEST, no la puedo ejecutar en una iteración, es grande porque involucra otras cosas…

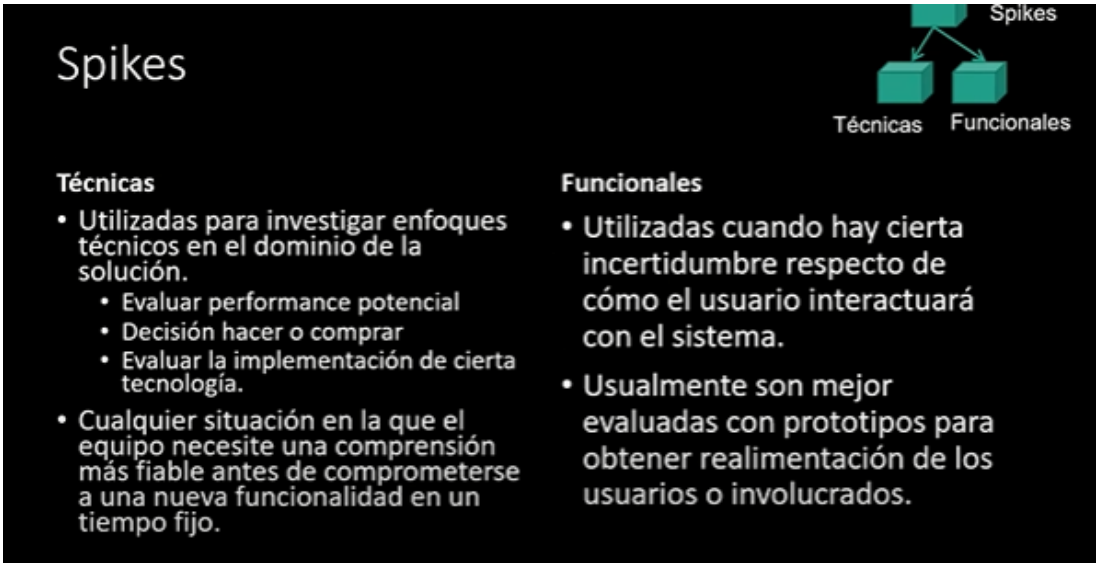
Theme - Tema → se usa como agrupador de US

representación de una iteración, como se trabaja con requerimientos ágiles



Spikes

- Tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una US u otra faceta del proyecto.
- Se clasifican en: técnicas y funcionales. Algunas US requieren de ambos tipos de spikes.
- Pueden utilizarse para:
    - Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.
    - Analizar un comportamiento de una historia compleja y poder así dividirla en piezas manejables.
    - ganar confianza frente a riesgos tecnológicos, investigando o prototipando para ganar confianza.
    - Frente a riesgos funcionales, donde no está claro como el sistema debe resolverla interacción con el usuario para alcanzar el beneficio esperado.



Lineamientos para Spikes

- Como es una US debe ser estimable, demostrable y aceptable .
- No toda pequeña incertidumbre justifica generar una spike, a veces se toma como parte de la US directamente. Cuando ya hay que capacitar el equipo, cuando hay complejidad alta, alinear el negocio e investigar lo que falte ahí sí justifica la Spike.
- La excepción, no la regla
    - Toda historia tiene incertidumbre y riesgos.
    - El objetivo del equipo es aprender a aceptar y resolver cierta incertidumbre en cada iteración.
    - Los spikes deben dejarse para incógnitas más críticas y grandes
    - utilizar spikes como última opción.
- Implementar la spike en una iteración separada(conviene en una iteración anterior a la que se está ejecutando) de las historias resultantes. Salvo que el spike sea pequeño y sencillo y sea probable encontrar una solución rápida en cuyo caso, spike e historia pueden incluirse en la misma iteración.

Recordemos lo que es requerimientos ágiles

Porque hay alta probabilidad de que cambie

Tips para que las US sean útiles para el equipo

- un paso a la vez(evitar la palabra "Y"). No mezclar 2 us, dividir.
- no olvidar la parte de la conversación: qué necesita y qué va a validar el PO, qué espera al final de la iteración
- usar palabras claras en los criterios de aceptación
- Las us se escriben desde la perspectiva del usuario
- no forzar todo para escribirlo como us

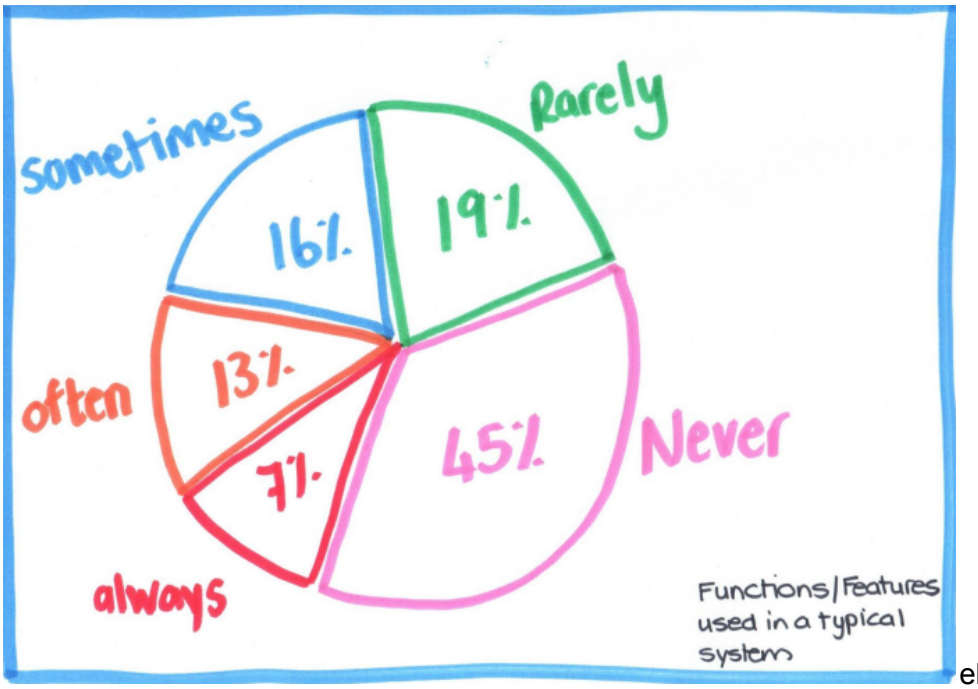
CLASE DE GESTIÓN DE PRODUCTOS

¿Por qué creamos productos?

- Para satisfacer a los clientes
- Para tener muchos usuarios logueados
- Para obtener mucho dinero
- Para realizar una gran visión, cambiar el mundo

¿Qué características realmente utilizamos del producto de software?

Tener features/funciones que no utilizamos es gastar plata innecesariamente. Hay que eliminar el desperdicio.



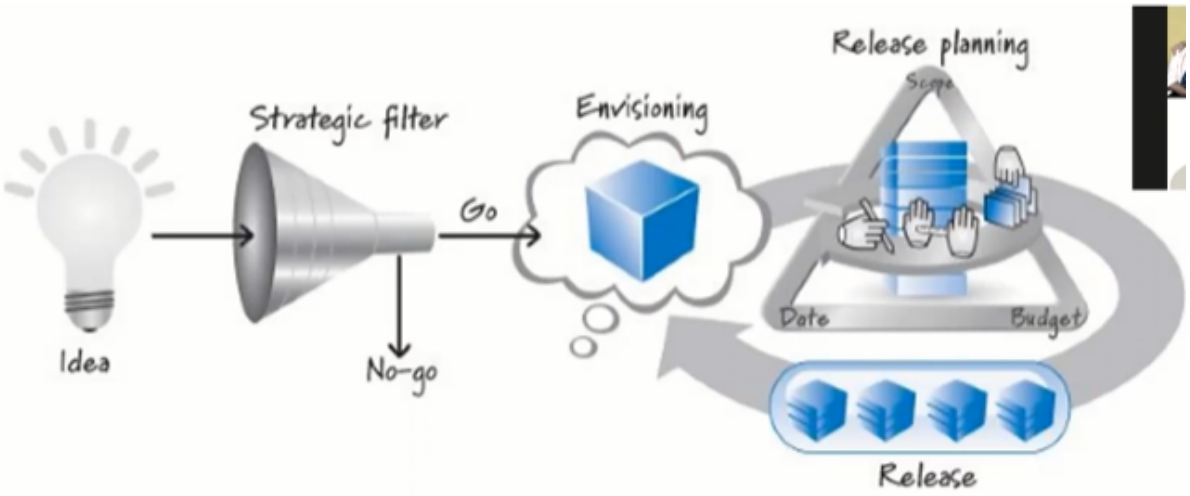
el 45% nunca se utilizan

Evolución de los productos de Software

La pirámide está en términos de necesidades. Si las de abajo (básicas) no se cumplen no hay fe en subir a las de más arriba.

¿Cómo podemos controlar el esfuerzo que realizamos? → para usar la cantidad de esfuerzo necesario (no de más) para cumplir los objetivos podemos utilizar la **técnica UVP** (visión del producto)

creación de productos: hay una idea que tiene alguien y se hace un filtro estratégico, tenemos que tener un sponsor, que nos apoye la idea, con aporte de dinero. Una vez que tengo el sponsor, empiezo a pensar cómo materializar la visión del producto, y como hago para que finalmente sea un producto de software que sea utilizado. Empiezo a plantear features, las voy priorizando y esta priorización cuales van al primer release del producto, cuales al segundo… etc.



Valor vs. Desperdicio

¿Cuales de nuestros esfuerzos crean valor y cuales son desperdicio? Lean thinking define la creación de valor como proveer beneficios a los clientes, cualquier otra cosa es desperdicio.
La productividad de un Startup no puede medirse en términos de cuánto se construye cada día, por el contrario se debe medir en términos de averiguar la cosa correcta a construir cada día

Algunos conceptos importantes para la Gestión de Productos:

- **Minimal Viable Product:** quiero definir el MVP para mostrarle a alguien el producto funcionando con las carca mínimas para saber si es aceptado o no y a partir de eso recibir una retroalimentación para ver si seguir evolucionandolo o ir para otro lado. Es el primer release, es tener un producto que me permita validar mi hipótesis (que se vea útil en cuanto al uso de los clientes)
- **Minimal Release Feature:** que tiene que tener mínimamente mi producto para poder convertirse en la primera release.
- **Minimal Marketable Feature:** tiene que ver con la parte e marketing, que es lo mínimo que tiene que tener mi producto para que la gente de marketing pueda venderlo

Metodología Startup

- trata de reducir tiempo y costo a la hora de crear empresas usando hipótesis e experimentación

- Creamos una hipótesis, un producto mínimo viable y métrica relevante (datos que nos fijamos para juzgar nuestra hipótesis). Luego se crea un bucle de feedback. Luego debemos pivotar con lo aprendido, modificando la hipótesis, pero si la hipótesis es buena avanzaremos más rápido invirtiendo mas rápido para crecer mas.

MVP

Se crea un **Producto Mínimo Viable (MVP)** para poder probar la hipótesis. Es solo para asegurarse de que sea viable como producto, ver si sirve o no y de ahí ver como seguir.

- MVP también es una hipótesis. Sirve para ver si los clientes están interesados en el producto que tengo para ofrecer.
- Podría ser lo suficientemente bueno para encontrar un mercado o no.
- El MVP se centra en un producto que no existe (no es lo mismo que un prototipo)
- Puede pasar que en lugar de validar la hipótesis, haya una característica del producto que MÁS le interesa al cliente. Ahí es cuando tengo que cambiar el foco, y puede convertirse en **MVP2**. Esto puede ocurrir repetidas veces.
- Usted puede crear un producto real que puede ofrecer a los clientes y observar su comportamiento real con el producto o servicio.
- Tiene el valor suficiente para que las personas estén dispuestas a usarlo o comprarlo inicialmente
- Demuestra suficiente beneficio futuro para retener a los primeros usuarios
- Proporciona un ciclo de retroalimentación para guiar el desarrollo futuro.

MMF → característica mínima comerciable, se descompone el producto y vemos qué features se pueden comercializar. Objetivo: aportar valor.

- En el caso de áreas con una fuerte indicación de valor, puede directamente definir un MMF (Características mínimas comercializables. Para encontrar la pieza mínima que pueda empezar a traer crecimiento.
- La razón para dividir una característica grande en MMF más pequeños es principalmente el tiempo de comercialización (Time to market) y la capacidad de aportar valor en muchas áreas.

MVF → característica Mínima Viable. No me centro en el producto viable. Si la MVF es exitosa puedo realizar más MMF en esa área para tomar ventaja. Sino, se puede cambiar a otro enfoque. El MVF es una versión mini del MVP.

- El producto se cultiva en mercados inciertos al intentar varios MVP.
- Cuando se logra ajustar el producto en el mercado de productos se combinan MMF y MVF según el nivel de incertidumbre del negocio / requisitos en las áreas en las que se está enfocando.
- Si bien los MVP / MMF / MVF son atómicos desde una perspectiva empresarial (no puede implementar y aprender de algo más pequeño) pueden ser bastante grandes desde la perspectiva de la implementación.

MVP

▪ Versión de un nuevo producto creado con el menor esfuerzo posible
▪ Dirigido a un subconjunto de clientes potenciales
▪ Utilizado para obtener aprendizaje validado.
▪ Más cercano a los prototipos que a una versión real funcionando de un producto.

MMF

▪ es la pieza más pequeña de funcionalidad que puede ser liberada
▪ tiene valor tanto para la organización como para los usuarios.
▪ Es parte de un MMR or MMP.

MMP

▪ Primer release de un MMR dirigido a primeros usuarios (early adopters),

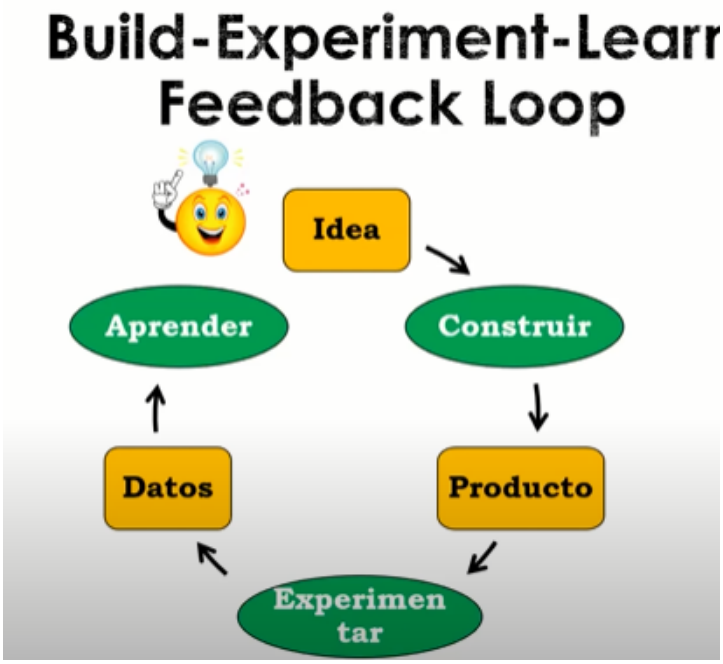
▪ Focalizado en características clave que satisfarán a este grupo clave.

MMR

▪ Release de un producto que tiene el conjunto de características más pequeño posible.
▪ El incremento más pequeño que ofrece un valor nuevo a los usuarios y satisface sus necesidades actuales.
▪ MMP = MMR1

MVP vs MMF o MMP: Errores comunes

- confundir un MVP, que se enfoca en el aprendizaje, con Característica Comerciable Mínima (MMF) o con Producto Comercializable Mínimo (MMP), ambos se enfocan en GANAR.
- El riesgo de esto es entregar algo sin considerar si es lo correcto que satisface las necesidades del cliente
- enfatizar la parte mínima de MVP con exclusión de la parte más viable. El producto entregado no es de calidad suficiente para proporcionar una evaluación precisa de si los clientes utilizarán el producto.
- Entregar lo que consideran un MVP, y luego no hacer más cambios a ese producto, independientemente de los comentarios que reciban al respecto



Hacer este feedback lo más rápido que se pueda para poder concretar la visión del producto definitivamente.

El éxito no es entregar un producto, el éxito se trata de entregar (o característica de producto) que el cliente usará.
La forma de hacerlo es alinear los esfuerzos continuamente hacia las necesidades reales de los clientes .

EL FOCO ESTÁ EN LA FASE DE EXPERIMENTACIÓN

Ejemplo de MVP → Dropbox

La fase construir: MVP

- ingresar lo más rápido posible con un producto mínimo viable MVP
- Un MVP varía en complejidad desde pruebas de humo (smoke tests) extremadamente simples (poco más que un anuncio) hasta prototipos tempranos.

Audacia de cero: aplazar el lanzamiento de cualquier versión de un producto hasta que esté seguro del éxito.

Supuestos “saltos de fe”

Los elementos más riesgosos del plan / concepto de una startup (es decir, las partes de las que todo depende) se denominan supuestos de salto de fe.

La mayoría de las personas no conocen una determinada solución (o incluso un problema); pero una vez que experimentan la solución, ¡no pueden imaginar cómo vivirían sin ella!

Supuestos de “Saltos de Fe”

Hipótesis del valor:
- Prueba si el producto realmente está entregando valor a los clientes después de que comienzan a usarlo
- Una métrica de prueba: tasa de retención

Hipótesis de crecimiento:
- Prueba cómo nuevos clientes descubrirán el producto
- Una métrica de prueba: tasa de referencia o Net Promoter Score (NPS)

PREPARAR UN MVP