

Prácticas ágiles

DIFERENTES FORMAS DE SER ÁGILES Y LLEVAR EL VALOR A LOS
CLIENTES

Entrega orientada al valor

En proyectos ágiles se busca entregar valor al cliente la más rápido posible. Para ello, en lugar de intentar entregar un producto final con todas las funcionalidades, se suele trabajar con entregas parciales de menor alcance. Por ejemplo:

- Producto viable mínimo (MVP)
- Características comercializables mínimas (MMF)

Producto Viable Mínimo (MVP)

El MVP es la versión preliminar de un nuevo producto que permite al equipo obtener retroalimentación de los clientes de manera anticipada para descubrir si están construyendo el producto correcto.



Mínimo

La base más rudimentaria y básica de la solución posible



Viable

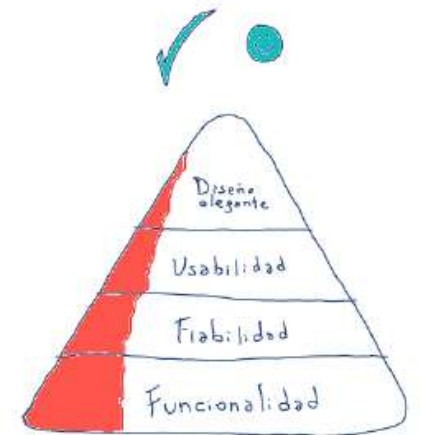
Suficiente para los primeros usuarios



Producto

Algo tangible que los clientes puedan tocar y sentir

El MVP no puede estar basado solamente en las funcionalidades del producto, sino que debe incluir también algo de fiabilidad, usabilidad y diseño.

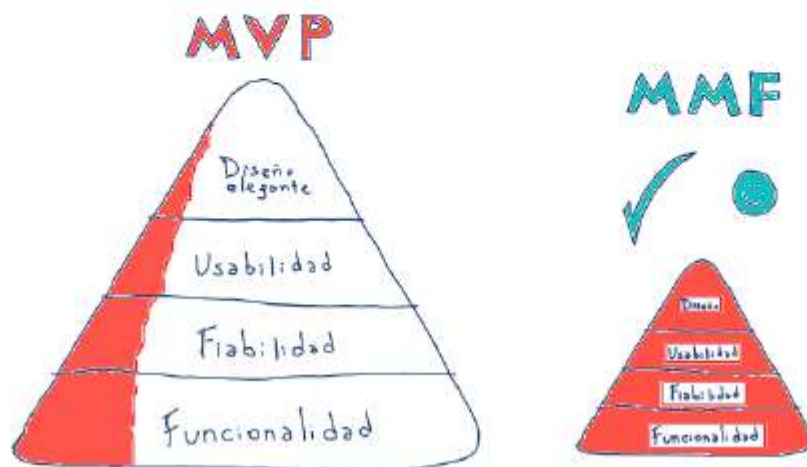


Característica comercializable mínima (MMF)

Otra forma de establecer prioridades sobre el alcance inicial del proyecto es definiendo las característica comercializable mínima (MMF: Minimum Marketable Feature).

Si bien los conceptos MVP y MMF se suelen utilizar como sinónimos, no significan lo mismo.

MMF es la unidad de funcionalidad más pequeña posible que proporciona un valor tangible a los clientes. A diferencia del MVP que podría ser un prototipo preliminar, MMF siempre es un producto terminado.

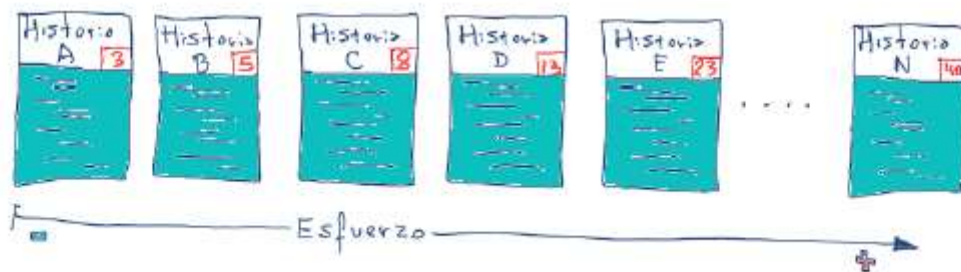


Un ejemplo de MMF es lanzar un celular con un software que incluye las principales funcionalidades y luego agregar gradualmente características adicionales durante la marcha mediante actualizaciones de ese software; en lugar de crear un producto masivo con varias características de una sola vez que luego no son valoradas por el cliente.

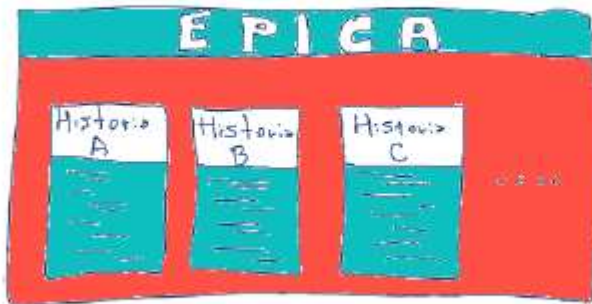


Priorización de los requisitos

El esfuerzo para realizar cada historia de usuario podría ser estimado en horas. Sin embargo, el equipo de desarrollo suele agregar puntos de historia (story-points) a cada historia de usuario según el esfuerzo relativo necesario para completarlas. Estos puntos de historia no tienen relación con horas de trabajo. Por ejemplo, a una historia de mínimo esfuerzo se le podrían asignar 3 puntos, otra historia de esfuerzo medio 23 puntos y una historia de bastante esfuerzo podría tener 40 puntos de historia.



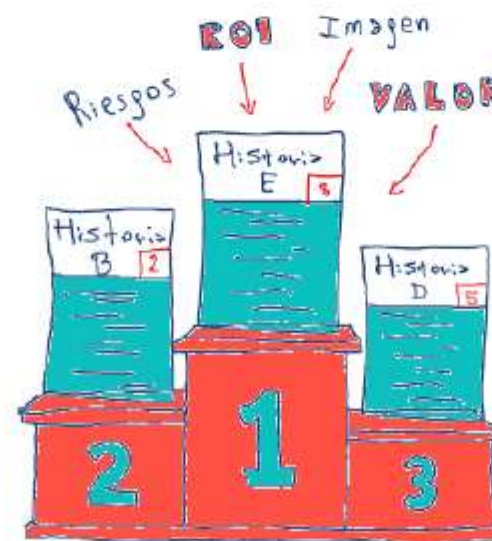
Las **épicas** son una gran historia de usuario que no puede completarse en una iteración. Por lo tanto, las épicas se dividen en historias de usuario más pequeñas las cuales podrán ser entregadas en distintas iteraciones.



Las **historias de usuario** son una breve descripción escrita que representa un requisito o funcionalidad del proyecto utilizando el lenguaje común del usuario.

El dueño del producto y las personas vinculadas al negocio son quienes conocen mejor las historias que aportarán más valor. Para estimar el valor de cada historia se tendrán en cuenta factores tales como:

- Generar un mayor **retorno de la inversión**
- Resolver más **problemas**
- Optimizar los **procesos** más importantes
- Resolver problemas **tecnológicos** para poder avanzar con el desarrollo del producto
- Reducir **riesgos**
- Mejoras de la **imagen** empresarial
- Incrementar la participación de **mercado**
- **Esfuerzo** necesario o costos asociados a cada historia



Técnicas cuantitativas de priorización de requisitos

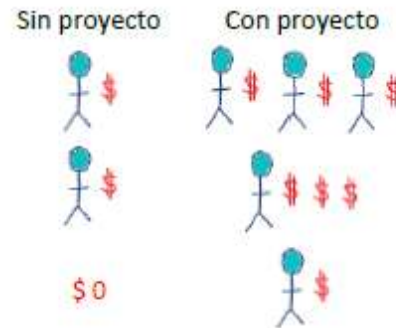
Retorno de la inversión (ROI)

Las entregas incrementales del proyecto por lo general van a generar algún tipo de ingresos o ahorro de costos a la organización. Los ingresos podrían clasificarse en:

Nuevos: obtenidos por nuevos clientes

Incrementales: obtenidos de los clientes existentes al agregar nuevas funcionalidades al producto

Retenidos: se perderán los clientes actuales si no se desarrolla una nueva característica



En la tabla a continuación se presenta un ejemplo con los ingresos generados por cinco épicas en las cuáles es necesario invertir \$100.

Flujo Fondos	Hoy	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Total	ROI
Épica A	-\$ 100	\$ 0	\$ 0	\$ 25	\$ 40	\$ 50	\$ 15	15%
Épica B	-\$ 100	\$ 55	\$ 40	\$ 5	\$ 5	\$ 5	\$ 10	10%
Épica C	-\$ 100	\$ 0	\$ 0	\$ 0	\$ 0	\$ 117	\$ 17	17%
Épica D	-\$ 100	\$ 20	\$ 20	\$ 20	\$ 20	\$ 25	\$ 5	5%

El **retorno de la inversión (ROI: return on investment)** del flujo de fondos de la Épica A sería de 15% en 5 meses (\$15 de ganancia neta / \$100 de inversión inicial).

$$ROI = \frac{\text{Ingresos netos}}{\text{Inversión}}$$

Valor actual neto (VAN)

Mientras antes se obtengan los ingresos, más beneficioso será para el proyecto.



Para calcular el valor actual de un ingreso futuro hay que considerar el valor del dinero en el tiempo. O sea, \$100 mañana valen menos que \$100 hoy. El costo de oportunidad del dinero podría ser el retorno de invertir en otros proyectos de riesgo similar.

Supongamos, por ejemplo, que el costo de oportunidad del dinero fuera de 1% mensual y que un lanzamiento del proyecto generará un valor de \$100.000 dentro de 3 meses. Ese dinero equivaldría a como si hoy hubiéramos tenido un ingreso de \$97.059.

$$\text{Valor Actual} = \text{Valor Futuro} / (1+i)^n = \$100.000 / 1,01^3 = \$97.059$$



Supongamos ahora que para desarrollar una épica tenemos que invertir \$100 hoy para obtener ingresos de \$30 en el mes 3, \$40 en el mes 4 y \$50 en el mes 5. Para saber si es rentable o no esa inversión, tenemos que calcular el valor actual neto.

	Hoy	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Total
Flujo Fondos	-\$ 100	\$ 0	\$ 0	\$ 30	\$ 40	\$ 50	\$ 20
Valor actual	-\$ 100	\$ 0	\$ 0	\$ 29,12	\$ 38,44	\$ 47,57	\$ 15,13

El valor actual neto de esa inversión es la sumatoria de todos los valores actuales, en este ejemplo \$15,13 (-100 + 29,12 + 38,44 + 47,57) utilizando una tasa de descuento del 1% mensual. Si el VAN es positivo, es rentable realizar esa inversión.

$$VAN > 0 \quad \checkmark$$

Técnicas cuantitativas de priorización de requisitos

Tasa Interna de Retorno (TIR)

Otra forma de priorizar las épicas sería comparando la **tasa interna de retorno (TIR)** de cada alternativa. La TIR es la tasa de interés que hace el valor actual neto igual a cero.

Siguiendo con el mismo ejemplo, en la tabla se presenta la TIR de cada alternativa:

Flujo Fondos	Hoy	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	TIR	Prioridad
Épica A	-\$ 100	\$ 0	\$ 0	\$ 25	\$ 40	\$ 50	3,38%	2º
Épica B	-\$ 100	\$ 55	\$ 40	\$ 5	\$ 5	\$ 5	5,62%	1º
Épica C	-\$ 100	\$ 0	\$ 0	\$ 0	\$ 0	\$ 117	3,19%	3º
Épica D	-\$ 100	\$ 20	\$ 20	\$ 20	\$ 20	\$ 25	1,60%	

No te preocupes por intentar calcular las TIR manualmente, ya que eso no lo harás jamás en la vida real y tampoco lo necesita para rendir la certificación PMI-ACP®. ¡Podrías utilizar la fórmula TIR en Excel y listo! 🟢

A los fines de demostrar que la TIR es la tasa de descuento que hace el VAN igual a cero, en la tabla a continuación descontamos el flujo de fondos de cada alternativa a las TIR.

Valor actual	Hoy	1	2	3	4	5	Total = VAN
Épica A	-\$ 100	\$ 0	\$ 0	\$ 22,63	\$ 35,02	\$ 42,35	\$ 0
Épica B	-\$ 100	\$ 52,07	\$ 35,86	\$ 4,24	\$ 4,02	\$ 3,80	\$ 0
Épica C	-\$ 100	\$ 0	\$ 0	\$ 0	\$ 0	\$ 100	\$ 0
Épica D	-\$ 100	\$ 19,69	\$ 19,38	\$ 19,07	\$ 18,77	\$ 23,10	\$ 0

Cuando la TIR es mayor que el costo de oportunidad del dinero, el VAN es positivo. En nuestro ejemplo, la TIR de A (3,38%), B (5,62%) y C (3,19%) son mayores a la tasa de descuento del 3%. Por lo tanto, el VAN de A, B y C son positivos.

$$TIR > i \rightarrow VAN > 0 \quad \checkmark \bullet$$

Por otro lado, si la TIR es inferior al costo del dinero, el VAN es negativo, como ocurre con la historia D (TIR 1,6% menor a tasa de descuento del 3%).

$$TIR < i \rightarrow VAN < 0 \quad \times \text{😞}$$

Si bien el ROI, VAN, o TIR deberían ser los criterios más utilizados de priorización, en la práctica suele ocurrir que no se cuenta con la información para poder estimar el flujo de fondos que podría generar cada épica o historia de usuario. Además, invertir demasiado tiempo en conseguir esa información podría ser poco eficiente en proyectos ágiles. Por lo tanto, a continuación, presentaremos otros métodos cualitativos de priorización.


Métodos cualitativos de priorización (1)

Método de los 100-puntos

El método de los 100 Puntos (o 100 dólares o Monopoly Money) se utiliza para priorizar las historias de usuario del listado de trabajo pendiente en función del valor que agregan al negocio.

A cada interesado se les da 100 puntos (o \$100) para que puedan distribuir entre las historias de usuario en función de lo que consideran que agregará más valor.

Por ejemplo, en la tabla a continuación, Roberto decidió que las cinco historias tienen igual importancia, asignando \$20 a cada una. María decidió que la historia A es lo más importante y que E no agrega valor. Por último, Nurg considera que B es lo más importante.

						TOTAL
 Roberto	\$20	\$20	\$20	\$20	\$20	\$100
 María	\$40	\$30	\$20	\$10	\$0	\$100
 Nurg	\$20	\$40	\$30	\$10	\$0	\$100
TOTAL	\$80 2°	\$90 1°	\$70 3°	\$40	\$20	



Cada persona distribuye sus \$100 entre las historias, luego se suman los valores de todos los participantes para obtener la lista de historias priorizadas. Siguiendo con el ejemplo, la historia de usuario más importante sería B, luego A y después C.

Análisis Kano

El modelo Kano es una teoría de desarrollo de productos y de satisfacción del cliente que clasifica a las preferencias del cliente en cinco categorías o atributos.



1. **Calidad requerida** (Must-be Quality): no generan satisfacción cuando se cumplen porque son atributos básicos, pero dan lugar a gran insatisfacción cuando no se cumplen.
2. **Calidad unidimensional** o de resultado: dan como resultado la satisfacción cuando se cumplen e insatisfacción cuando no se cumplen porque fueron ofrecidos por el vendedor.
3. **Calidad de entusiasmo** o atractiva (Delighters): proporcionan satisfacción cuando se logran plenamente, pero no causan insatisfacción cuando no se logran porque no son esperados por el cliente ni fueron mencionados antes de la compra.
4. **Calidad indiferente:** aspectos que no son ni buenos ni malos; no generan satisfacción o insatisfacción del cliente.
5. **Calidad inversa:** un alto grado de atributos resulta en la insatisfacción de algunos clientes. Por ejemplo, algunos clientes prefieren los productos de alta tecnología con muchas funcionalidades, mientras que otros quedan insatisfechos si un producto tiene demasiadas funcionalidades tecnológicas que les complican el uso.

Con el modelo Kano clasificamos los requisitos según los tipos de calidad para luego priorizarlos. Por ejemplo, la calidad requerida y unidimensional suele ser lo más prioritario.

Métodos cualitativos de priorización (2)

MoSCoW

Categorizar los diferentes requisitos del proyecto desde los que son estrictamente necesarios (prioritarios) hasta los que no serían necesarios por el momento (no prioritarios). Las letras MoSCoW se derivan de los siguientes términos del inglés:

- M**ust have (obligatorio)
- S**hould have (deberían estar)
- C**ould have (podría implementarse)
- W**on't have (no lo queremos por el momento)

Puño a cinco (fist to five)

El método de votación puño a cinco es una técnica utilizada para encuestar a los interesados sobre el valor de las historias de usuario a los fines de lograr un consenso en la priorización.

Cada miembro del equipo responde levantando su mano con un puño o con el número de dedos según el nivel de apoyo.



Puño: desea bloquear la propuesta porque cree que es perjudicial.



Un dedo: tiene reservas serias, pero no bloqueará la iniciativa. Se compromete a abrir la comunicación con respecto a sus reservas.



Dos dedos: tiene reservas importantes, pero apoyará la iniciativa.



Tres dedos: apoya la iniciativa.



Cuatro dedos: brinda un fuerte apoyo y participación, pero no está dispuesto a liderar la iniciativa.



Cinco dedos: lo considera como la mejor idea, tomará la iniciativa si se lo piden.

Métodos cualitativos de priorización (3)

Modelo de Karl Wieggers

El modelo de priorización de Karl Wieggers, utiliza una escala cualitativa del 1 al 9 para calificar cada uno de los siguientes factores: beneficio, penalidad, costo y riesgo.

- **Beneficio.** 1 sería que no genera ingresos o no está alineado con los requisitos comerciales del producto, hasta 9 con gran satisfacción del cliente o ROI.
- **Penalidad.** 1 si no hay penalización y 9 indica un problema muy serio. Por ejemplo, el incumplimiento de una regulación gubernamental podría incurrir en una multa alta, incluso si el beneficio para el cliente es bajo.

VALOR = Beneficio + Penalidad. Por ejemplo, los requisitos que tienen un beneficio y penalidad bajos agregan poco valor, por lo que pueden ser casos de “enchapado en oro”.

- **Costo:** gastos o esfuerzo de realizar el desarrollo. 1 implica bajo costo y 9 alto costo.
- **Riesgo:** amenazas o problemas tecnológicos de implementar la funcionalidad. 1 es de muy bajo riesgo y 9 muy alto riesgo.

COSTOS = Costo + Riesgo

Los ítems prioritarios son aquellos de alto valor y bajos costos. Una forma de calcular la prioridad es comparando el valor con los costos.

$$\text{Prioridad} = \text{Valor} / \text{Costos}$$

Veamos un ejemplo en la tabla a continuación donde los expertos comerciales y el equipo técnico han calificado con puntajes cualitativos del 1 al 9 cuatro historias de usuario.

Historia	Beneficio (a)	Penalidad (b)	Valor (c) (a) + (b)	Costo (d)	Riesgo (e)	Costos (f) (d) + (e)	Prioridad (c) / (f)
A	7	4	11	9	7	16	0,69
B	2	1	3	1	1	2	1,50
C	5	3	8	2	1	3	2,67
D	9	8	17	7	8	15	1,13

En este ejemplo las prioridades de desarrollo serán: 1º C, 2º B, 3º D y 4º A.

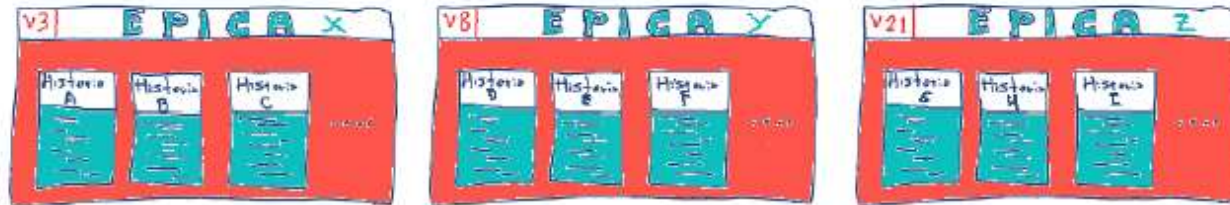
También se suelen trabajar las prioridades colocando ponderaciones. Por ejemplo, otorgar más ponderación a los beneficios que a la penalidad o más ponderación al costo que al riesgo.

Métodos cualitativos de priorización (4)

Puntos de valor de negocio

Así como se suelen asignar puntos de historia al esfuerzo relativo que requiere realizar cada épica o historia de usuario, se pueden asignar “puntos de valor de negocio (business value points)” relativos para cada épica.

Por ejemplo, a una épica de poco valor se le podrían asignar 3 puntos, otra épica de valor medio 8 puntos y una épica de mucho valor podría tener 21 puntos.



La priorización entre épicas se puede obtener comparando la relación entre los puntos de valor relativo con los puntos de esfuerzo relativo de cada épica.

$$\text{Priorización} = \text{Puntos de valor} / \text{Puntos de esfuerzo}$$

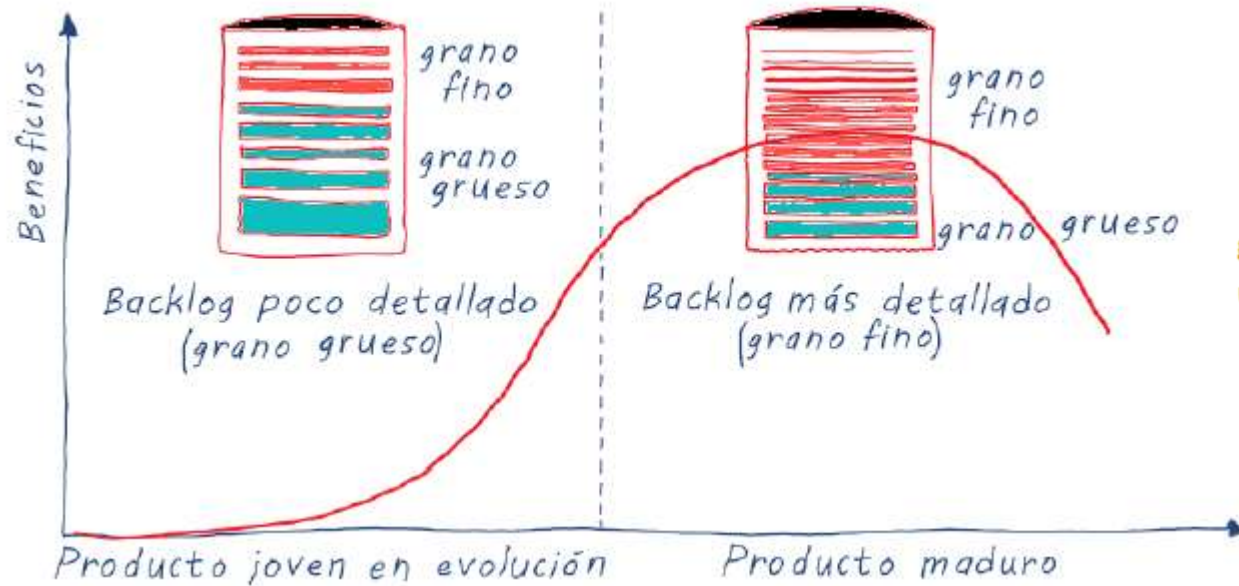
Al momento de priorizar, podríamos ponderar como más importante el valor que el esfuerzo.

En la tabla a continuación se presenta un ejemplo donde se tiene que priorizar entre siete épicas. El dueño del producto agregó puntos de valor relativo a cada épica y el equipo de desarrollo estimó el esfuerzo relativo de cada una de esas épicas. Además, se decidió ponderar en 70% los puntos de valor y en un 30% los puntos de esfuerzo.

Épica	Valor	70% Valor	Esfuerzo	30% Esfuerzo	V / E	Prioridad
A	2	1,4	5	1,5	0,93	6º
B	3	2,1	5	1,5	1,40	5º
C	5	3,5	2	0,6	5,83	2º
D	8	5,6	13	3,9	1,44	4º
E	13	9,1	5	1,5	6,07	1º
F	21	14,7	21	6,3	2,33	3º

Mientras mayor sea el ratio Valor / Esfuerzo, mayor será la prioridad. O sea, realizar primero las actividades de mayor valor para la menor cantidad de esfuerzo posible.

Especificación de los requisitos



Un product backlog excesivamente detallado o de grano fino (fine grained) será difícil de gestionar. Pero si es demasiado agregado o de grano grueso (coarse grained), tampoco será útil porque proporciona muy poca dirección al equipo de desarrollo.

Especificación mediante ejemplos (SBE: Specification by Example): enfoque colaborativo para definir los requisitos y las pruebas funcionales basado en ejemplos realistas, en lugar de enunciados abstractos.

Entregas rápidas y flexibles

Todo proyecto ágil persigue el triple objetivo de ser “rápido, flexible y fluido”.



Rápido (Fast): para que los entregable lleguen al cliente lo más rápido posible es necesario eliminar los desperdicios, las burocracias y la sobreproducción trabajando con lotes pequeños en ciclos incrementales de desarrollo manteniendo relaciones estrechas entre los interesados.



Flexible: el desarrollo del producto debe permitir la customización según las necesidades del cliente para poder gestionar los cambios que sean necesarios.

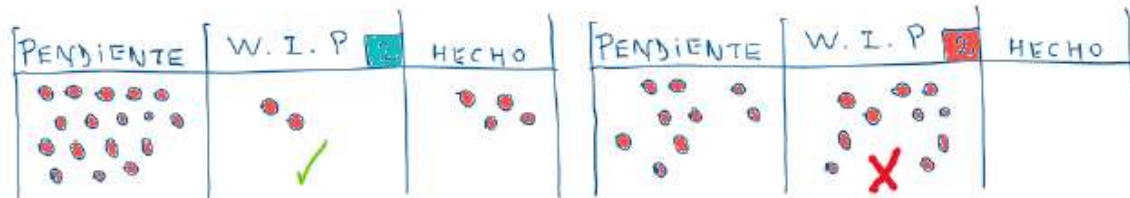


Fluido (Flow): entregar incrementos del producto de manera continua.

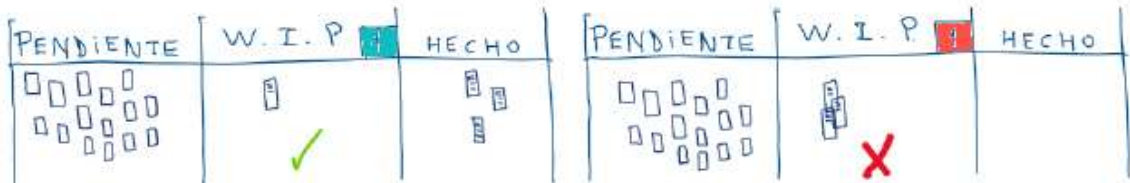
Una de las formas de realizar entregas **rápidas y fluidas** es aplicando el principio Lean que dice “que el valor fluya sin interrupciones”. Para ello será fundamental comprender el concepto de tiempo valor agregado y eliminar cualquier desperdicio que afecte el flujo de producción.

Trabajo en progreso (WIP: work in progress) (1)

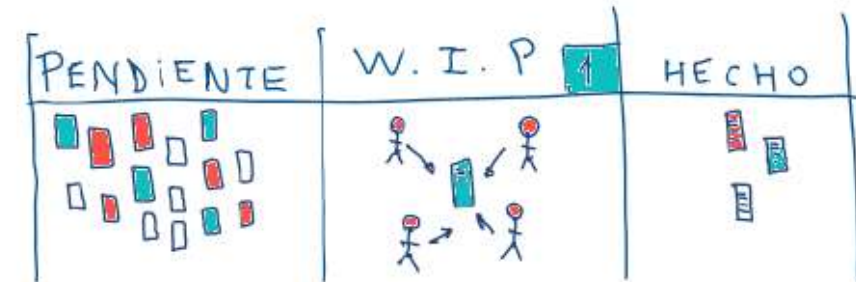
Por ejemplo, supongamos que una persona nos lanza pelotitas de ping pong y nuestro trabajo es atraparlas y colocarlas dentro de una caja. Si nos lanza una pelotita a la vez podríamos guardar una pelotita por segundo; si somos muy dúctiles tal vez podríamos atrapar hasta un máximo de 2 pelotitas (una con cada mano) y guardarlas en un segundo; pero si nos lanzan 10 pelotitas juntas, lo más probable es que se nos caigan casi todas, demoremos mucho tiempo en recogerlas, tengamos que volver a comenzar el trabajo y aumentar bastante el plazo de producción, donde el guardado de cada pelotita podría extenderse a minutos. O sea, en este ejemplo, el máximo WIP sería de 2.



Otro ejemplo podría ser el de una impresora, donde el máximo WIP de páginas en impresión sería 1 para imprimirlas lo más rápido posible. Si queremos imprimir 2 o más páginas al mismo tiempo, el papel se atascaría y demoraríamos más tiempo corrigiendo los problemas.



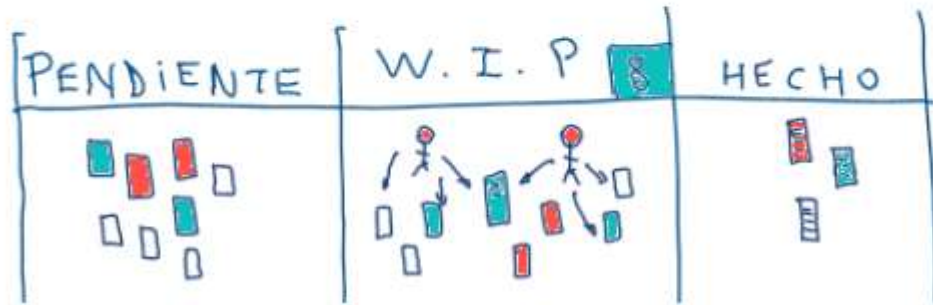
Si el límite del WIP fuera demasiado bajo, por ejemplo, un WIP de 1 para un equipo de 4 personas, esa tarea se podría realizar muy rápido ya que tendrá 4 personas trabajando solamente en esa actividad hasta terminarla.



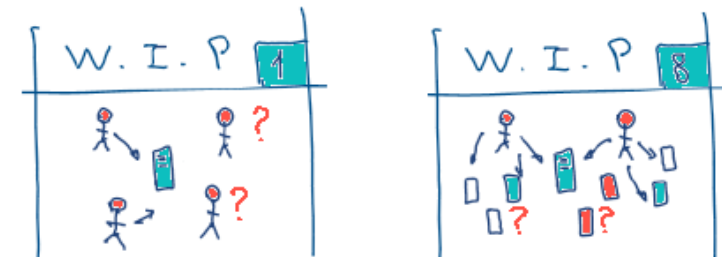
Si bien un WIP bajo fomenta la colaboración y trabajo en equipo, es probable que tengamos miembros del equipo ociosos porque no podrán trabajar juntos todo el tiempo sobre esa misma tarea. Además, cuando el WIP es muy bajo, tendremos varias tareas en estado de “pendientes” demasiado tiempo, en espera de que el equipo comience a trabajar sobre ellas. Por lo tanto, el “tiempo de entrega” desde que el cliente realiza el pedido hasta que se entrega el producto, será mayor de lo necesario.

Trabajo en progreso (WIP: work in progress) (2)

En el otro extremo, si el límite del WIP es demasiado alto, por ejemplo, un WIP de 8 para un equipo de 2 personas, existe el riesgo de empezar muchas tareas y terminar pocas. Frente a cualquier contratiempo en una tarea, el equipo, en lugar de enfocarse en resolver el problema y terminar la tarea, podrán optar por comenzar con otra tarea y demorar las que presenten problemas.



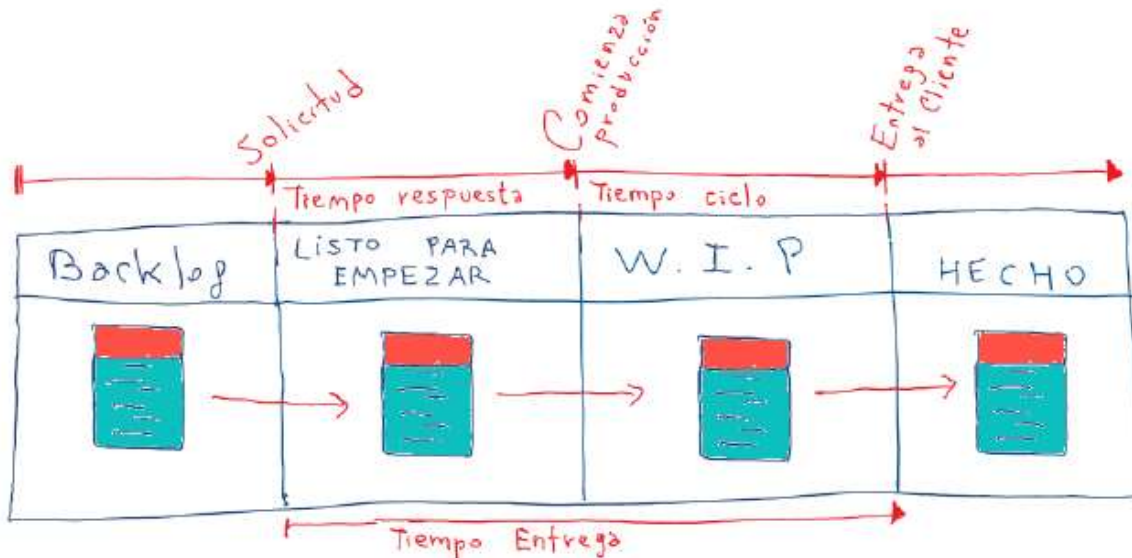
Resumiendo, si observamos miembros del equipo ociosos sin estar trabajando en tareas es probable que el WIP sea demasiado bajo. Y si observamos tareas en el listado del WIP pero que no tienen personas trabajando en ellas, es probable que el WIP sea demasiado alto.



Al aplicar los límites al WIP, el equipo tendrá la oportunidad de localizar los cuellos de botella en los procesos de trabajo antes de que éstos se conviertan en bloqueos que retrasan la entrega de valor al cliente.

Tiempos de la agilidad

- **Tiempo de respuesta (response time)**: desde que el cliente realiza una solicitud hasta que los miembros del equipo comienzan con el trabajo.
- **Tiempo de ciclo (cycle time)**: desde que el trabajo comienza hasta que el producto se entrega al cliente listo para su uso. Este tiempo de ciclo podría dividirse en el que agrega valor y en aquel que no agrega valor (ej. esperas, ineficiencias, etc.)
- **Tiempo de entrega (lead time)**: desde que el cliente realiza la solicitud hasta que finaliza la entrega del producto. O sea, la sumatoria del tiempo de respuesta más el tiempo de ciclo.



- **Tiempo ritmo (takt time)**: ritmo necesario para satisfacer la demanda del cliente. Por ejemplo, si el cliente quiere que se completen 30 puntos de historia en 15 días, el ritmo de producción promedio debería ser de 2 puntos de historia por día.

Ritmo = Cantidad de productos que se quieren / Tiempo disponible

- **Rendimiento (throughput)**: capacidad promedio de producción de los miembros del equipo. Por ejemplo, completar 5 puntos de historia promedio por día.

John Little definió la relación entre el tiempo de ciclo, WIP y rendimiento:

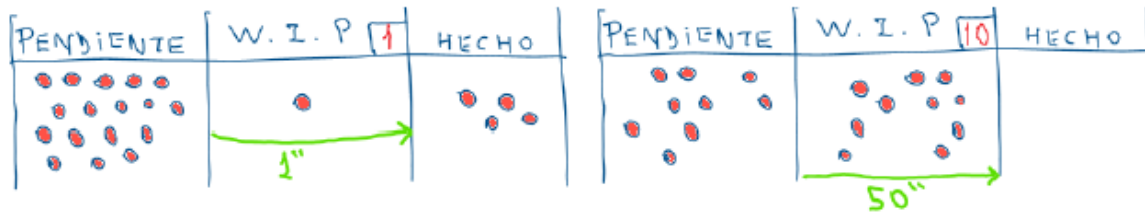
$$\text{Tiempo de Ciclo} = \frac{\text{WIP}}{\text{Rendimiento}}$$

O sea, si queremos acortar el tiempo de ciclo para entregar valor al cliente más rápido, deberíamos disminuir el WIP y/o aumentar el rendimiento.

$$\begin{aligned} \downarrow \text{WIP} &\Rightarrow \downarrow \text{Tiempo Ciclo} \\ \uparrow \text{Rendimiento} &\Rightarrow \downarrow \text{Tiempo Ciclo} \end{aligned}$$

Tiempos de la agilidad

Volvamos al ejemplo simple de que tenemos que guardar 10 pelotitas de ping pong en una caja. Si el WIP es limitado a un máximo de una pelotita, podríamos guardar 1 pelotita por segundo. Pero si nos lanzan 10 pelotitas juntas, se nos van a caer, tendremos que levantarlas y el tiempo promedio de producción podría elevarse a 5 segundos por pelotita, lo que es lo mismo a decir que el rendimiento bajaría a 0,2 pelotita por segundo.



Tiempo de ciclo ineficiente = 10 pelotitas / 0,2 pelotitas por segundo = 50 segundos

Tiempo de ciclo eficiente = 1 pelotita / 1 pelotita por segundo = 1 segundo

Por lo tanto, cuando el WIP es alto, para completar 10 pelotitas necesitamos un tiempo de ciclo de 50 segundos. Mientras que con un WIP bajo, podemos completar 1 pelotita por segundo (o 10 pelotitas en 10 segundos).

Una de las herramientas utilizadas para medir el tiempo de entrega y el tiempo de ciclo para varias tareas a través del tiempo es el diagrama de flujo acumulativo.

*Es preferible tener pocas tareas terminadas
en lugar de muchas tareas sin terminar*

¿Cómo logramos reducir el tiempo de ciclo y entrega?

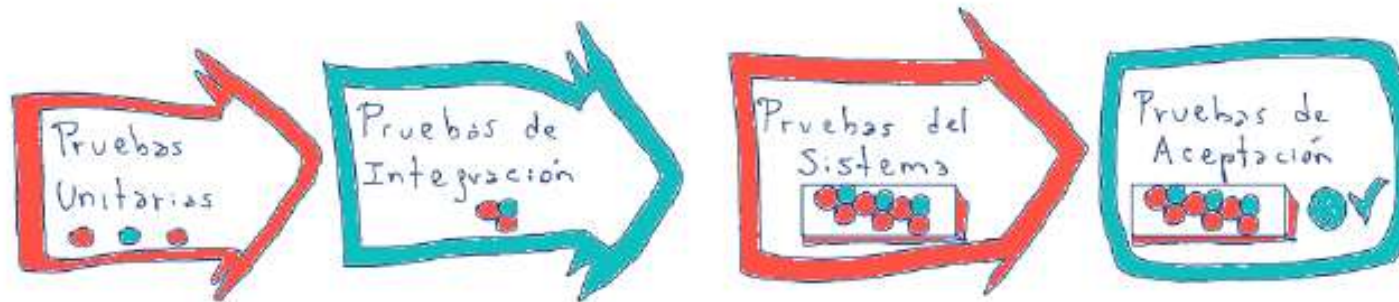
Podemos concluir que el tiempo de ciclo y entrega son estadísticas poderosas que proporcionan información valiosa sobre el flujo de trabajo, ayudan a identificar cuellos de botella y a comprender la capacidad de trabajo del equipo. Estos tiempos se pueden mejorar:

- Trabajando con solicitudes relativamente pequeñas para gestionarlas de manera ágil
- Reduciendo la cantidad de tareas que se encuentran en proceso (WIP)
- Mejorando el rendimiento (ej. reduciendo los errores y retrabajo)
- Reasignando los recursos según existan cuellos de botella o capacidad ociosa

Entregas de calidad

Pruebas en el desarrollo de software

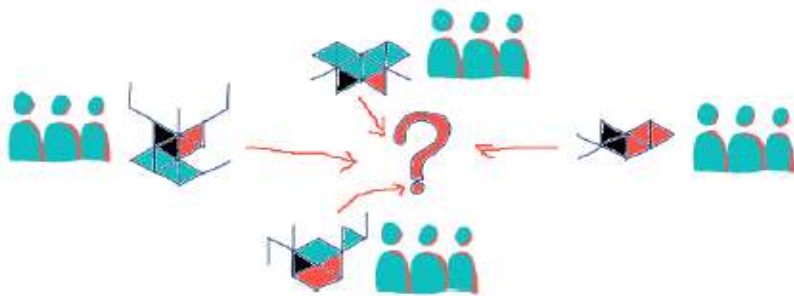
Por lo general, durante el proceso de desarrollo de software se suelen realizar como mínimo cuatro pruebas para asegurar la calidad del producto final.



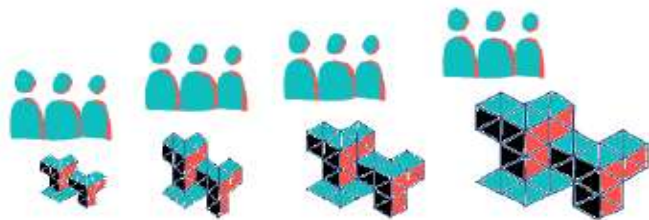
Integración Continua Sincrónica

Integración continua sincrónica

La integración de un software plantea problemas cuando diferentes equipos diseñan y programan diferentes partes de una aplicación. Sobre el final del proyecto, cuando se intentan compilar todas las partes para terminar el producto, aparecen los problemas. Esto ocasiona en varias oportunidades una larga fase de desarrollo para corregir las fallas, unir los fragmentos de código, realizar las pruebas y lograr que el sistema funcione correctamente.



Para mitigar estos inconvenientes, la integración continua consiste en hacer integraciones automáticas (compilación y ejecución de pruebas) varias veces al día para detectar fallos cuanto antes. Por ejemplo, luego de cada hora de desarrollo se descargan las fuentes desde el control de versiones, se compilan, ejecutan y generan informes.



Ventajas



Los desarrolladores pueden detectar y solucionar problemas de integración de forma temprana y continua, evitando el caos de última hora



Disponibilidad constante de una versión para pruebas, demos o lanzamientos anticipados



Ejecución inmediata de las pruebas unitarias



Monitorización continua de las métricas de calidad del proyecto

Desventajas



Difícil de mantener. Por ejemplo, crear un repositorio de código automatizado implica que los equipos deben crear el conjunto de pruebas adecuado y dedicar tiempo a escribir casos de prueba en lugar de desarrollar código.



Puede derivar en demoras cuando varios desarrolladores intentan integrar sus códigos al mismo tiempo.

Desarrollo guiado por pruebas (TDD)

El desarrollo guiado por pruebas (TDD: Test-driven development) es una práctica de ingeniería de software que involucra dos prácticas:

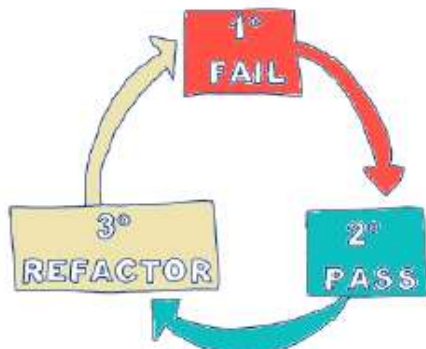
Escribir las pruebas primero (Test First Development): seguir el ciclo “Escribir una prueba automatizada fallida -> Ejecutar la prueba fallida -> desarrollar código para hacer pasar la prueba -> ejecutar prueba -> repetir”.

Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test) donde el desarrollador prueba cada componente individual del producto.

Refactorización (Refactoring): eliminar códigos duplicados o innecesarios

El proceso resumido de TDD, también conocido como red-green-refactor, sería el siguiente:

- 1º Escribir una prueba unitaria y verificar que falle (**Red**)
- 2º Escribir el código que hace que la prueba pase satisfactoriamente (**Green**)
- 3º Refactorizar el código escrito para limpiar códigos duplicados o innecesarios (Refactor)



En TDD los requisitos se traducen en pruebas, de este modo, cuando las pruebas pasan se garantiza que el software cumple con los requisitos que se han establecido.

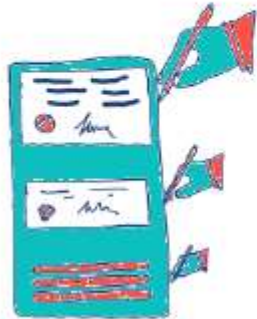
Veamos un ejemplo más completo de como sería el proceso de TDD

1. El cliente escribe su historia de usuario.
2. El equipo de proyecto escribe junto con el cliente los criterios de aceptación de esa historia, desglosándolos en pequeños criterios para simplificarlos la máximo posible.
3. Se escoge el criterio de aceptación más simple y se escribe en una prueba unitaria.
4. Se comprueba que la prueba falle.
5. Se escribe el código que hace pasar la prueba.
6. Se ejecutan todas las pruebas automatizadas para comprobar su funcionamiento
7. Se refactoriza y se limpia el código.
8. Se vuelven a pasar todas las pruebas automatizadas para comprobar que todo sigue funcionando.
9. Volvemos al punto 3 con los criterios de aceptación que falten y repetimos el ciclo una y otra vez hasta completar nuestra aplicación.

Contratos Ágiles

Contrato de varios niveles

Separar los elementos más cambiantes de un contrato de todo el resto, para permitir mayor flexibilidad y velocidad de contratación. Por ejemplo:



Nivel 1 - Los elementos comunes que no cambiarán como garantías, arbitraje, etc. se redactan en un acuerdo marco.

Nivel 2 - Los elementos sujetos a cambios (ej. tarifas, descripción del producto, etc.) se redactan en otro acuerdo.

Nivel 3 - Los elementos más dinámicos como el alcance, el cronograma y el presupuesto se formalizan en un acuerdo más liviano.

Alcance dinámico

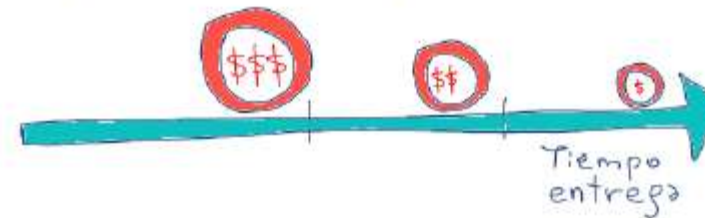
El cliente puede variar el alcance del proyecto en puntos específicos acordados con el proveedor manteniendo un contrato de precio fijo. Por ejemplo, las horas de programadores que se iban a utilizar para hacer la historia X, se pueden reasignar para desarrollar la historia Z que requiere un esfuerzo similar.



Descomponer el alcance en pequeños entregables (ej. historias de usuario) de precio fijo para que el cliente seleccione que entregable quiere que desarrollen sabiendo con anticipación el costo que deberá pagar por cada uno de ellos.



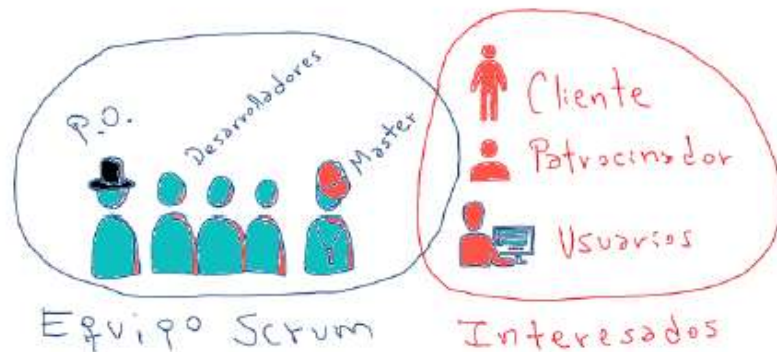
El proveedor cobra una tarifa horaria más alta cuando entrega antes de lo acordado y cobra una tarifa menor en caso de demora en la entrega.



Involucrar a los interesados

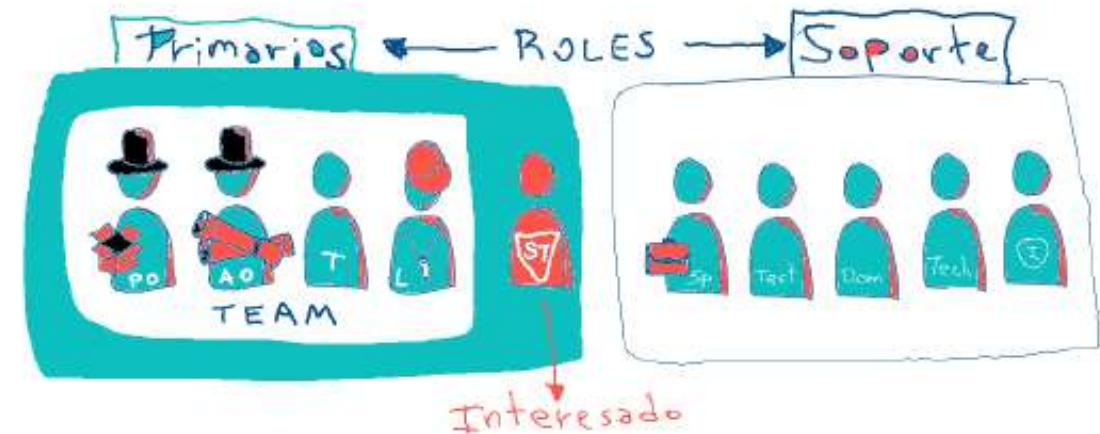
En la gestión tradicional de proyectos con ciclos predictivos, se entiende por interesados a todas aquellas personas u organizaciones que se verán afectadas por el proyecto. Por ejemplo: patrocinador, director del proyecto, miembros del equipo, proveedores, clientes, comunidad, etc. Sin embargo, en el ambiente de proyectos ágiles los miembros del equipo no forman parte de la definición de interesados.

Por ejemplo, en Scrum un interesado es cualquier persona con interés o influencia sobre el producto que no forma parte del equipo Scrum (dueño del producto, scrum master, equipo de desarrollo). Ejemplo de interesados en Scrum sería el cliente, el patrocinador y los usuarios.



El dueño del producto suele ser quien representa a los interesados en el equipo Scrum.

Por su parte, en Disciplined Agile los "interesados" no forman parte del equipo principal (dueño del producto, dueño de arquitectura, equipo de desarrollo, líder del equipo), ni del equipo de soporte (especialista, probador, experto de dominio, experto técnico, integrador).



Los interesados son personas u organizaciones que frecuentemente interactúan con el equipo para facilitar la creación de los productos influyendo durante todo el desarrollo del proyecto.

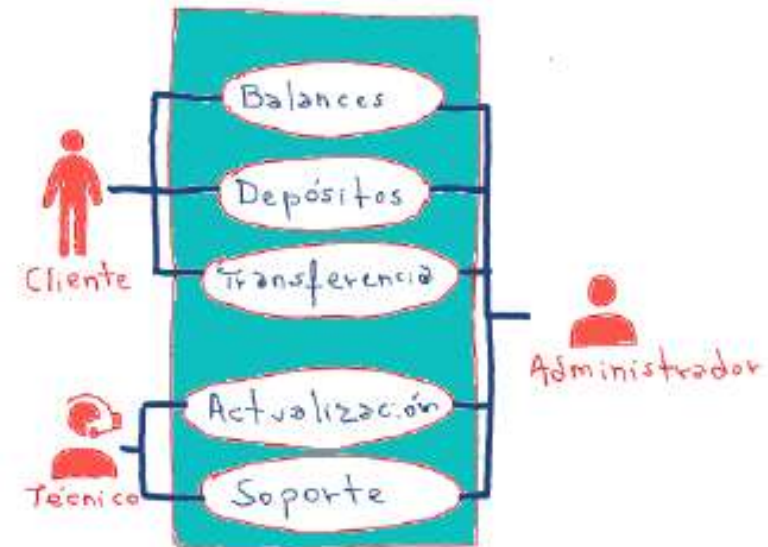
Diagrama del caso de uso

Los usuarios suelen formar parte de los interesados clave del proyecto, por lo tanto, es fundamental involucrarlos activamente en el proyecto.

Una herramienta para visualizar los usuarios de un sistema es el diagrama del caso de uso. Los tres elementos de este diagrama son:

- los usuarios
- los casos de uso
- las interrelaciones

Ejemplo de diagrama de caso de uso



El **caso de uso** es la descripción escrita de una funcionalidad del sistema que puede contener varias historias de usuario y puede requerir de varias iteraciones. Describe desde el punto de vista de un usuario, el comportamiento de un sistema cuando responde a una solicitud.

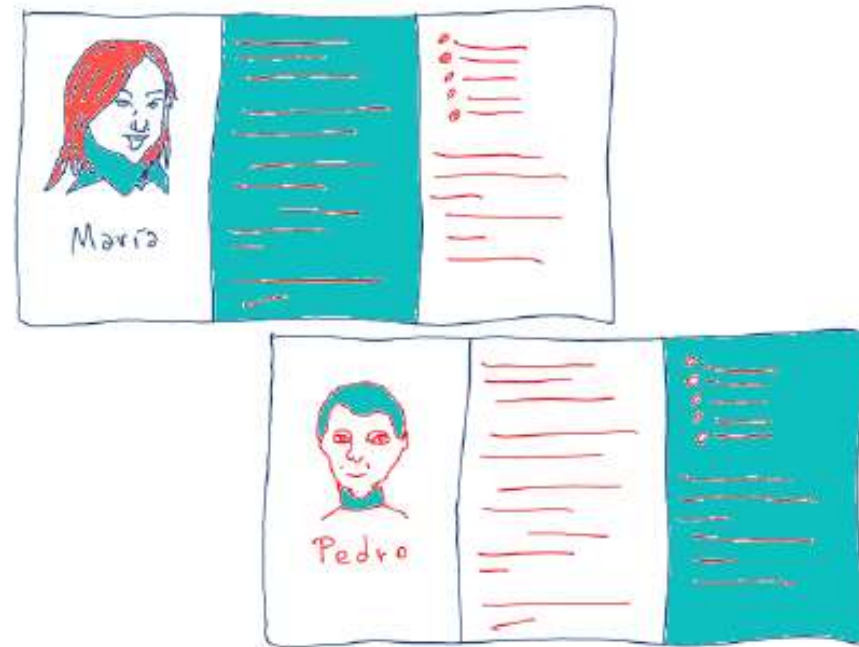
El diagrama del caso de uso describe cómo un usuario usa un sistema para lograr un objetivo particular.

Personas

Si bien los usuarios del sistema son interesados claves del proyecto, hay veces que no se tiene un acceso fácil a ellos para poder realizar el desarrollo y pruebas del producto antes de su lanzamiento. En estos casos se puede trabajar con el concepto de “persona”.

Una “persona” representa a un individuo ficticio, que se basa en usuarios reales, que interactúa con el sistema que se está desarrollando.

Cada persona se describe en una ficha con su nombre real, una imagen de rostro, su personalidad, motivaciones, edad, hobbies, experiencia laboral, familiares, etc.



Para describir personas de manera efectiva, hay que investigar y comprender a los usuarios reales. Las personas ayudan a guiar las decisiones sobre la funcionalidad y diseño del producto.

No deberíamos “inventar” personas, sino que la persona se descubre como resultado de una investigación para entender cuáles son las necesidades reales del usuario.

Acciones para involucrar a los interesados

1. Fomentar la **participación temprana**



Involucrar a los interesados en las fases iniciales del desarrollo del producto es fundamental para comprometerlos con el proyecto. Por ejemplo, invitar a los interesados clave en las discusiones preliminares sobre los requisitos o la definición de las historias de usuario.

2. Explicar los **beneficios globales** del producto



Explicar todos los beneficios que obtendremos con el producto aumenta el sentido de pertenencia sobre los impactos positivos del proyecto. Por ejemplo, no sólo explicar los beneficios directos que obtendrá la organización o los usuarios, sino los beneficios indirectos que obtendrá la comunidad al resolver un problema, mejorar su calidad de vida, etc.

3. Incluirlos en las **prioridades** del proyecto



Si bien el dueño del producto es el responsable final de establecer las prioridades de las historias de usuario a desarrollar, es fundamental mantener una comunicación fluida con los interesados para que ellos provean de información antes de decidir sobre las prioridades.

4. Involucrarlos en la **planificación** del lanzamiento



Si bien los miembros del equipo de desarrollo son los principales responsables de planificar las tareas necesarias a desarrollar durante las iteraciones antes de un lanzamiento, participar a los interesados será fundamental para aportar información sobre dependencias, riesgos, oportunidades, recursos compartidos, valor, etc.

5. Solicitar **comentarios** durante las revisiones de la iteración



Invitar a los interesados durante las reuniones de revisión de la iteración o revisión del producto para escuchar su retroalimentación es una excelente práctica para mantenerlos involucrados con el proyecto y mejorar la calidad.

Visión compartida (1)

Acta de constitución del proyecto



El acta de constitución (o carta del proyecto) de un proyecto ágil es un resumen de alto nivel elaborado por el equipo que incluye la visión, misión, factores clave de éxito y los acuerdos entre el equipo y los interesados externos. Este acta es imprescindible para ayudar a crear una comprensión común del proyecto entre los interesados.

Suele tener el tamaño máximo de una hoja de rotafolio para que esté visible en la sala del equipo si están colocados, o una hoja en formato electrónico si trabajan de manera remota.

Este documento se puede utilizar para iniciar el proyecto ágil formalmente.

No se incluye información adicional como hitos, supuestos y restricciones que suelen formar parte del acta de constitución de proyectos de ciclos predictivos.

El acta de constitución ágil se enfoca más sobre "Cómo" se resolverá un problema o se gestionará el proyecto en lugar de "Qué" se desarrollará exactamente para resolver ese problema. De esta forma no se imponen límites innecesarios al alcance del proyecto.

Al principio del proyecto sabemos poco del producto o servicio final, por lo tanto, tendremos una versión resumida del acta que incluye solamente:

- **Visión** - ¿Por qué existe el proyecto?
- **Misión** - ¿Cómo va a trabajar el equipo para lograr la visión?
- **Criterios de éxito** - ¿Qué se logrará con el proyecto?



Cuando el proyecto está más avanzado, se podría agregar en este documento de una sola página información adicional como:

- Antecedentes
- Objetivos
- Interesados

Asegurar que los contenidos del acta de constitución sean conocidos y aprobados por todos los miembros del equipo será fundamental para que se involucren con el proyecto.

La visualización del proyecto podría cambiar a medida que se recopila nueva información, por lo que el acta de constitución deberá revisarse periódicamente para garantizar una comprensión clara de los interesados.

Visión compartida (2)

Caja de visión del producto

La caja de visión del producto (o arquitectura del proyecto) es una herramienta para simplificar la visualización del producto condensando la información en un espacio limitado.

Para realizar la caja de visión se suele realizar un taller práctico colaborativo con los miembros del equipo e interesados clave antes de comenzar con un nuevo proyecto. En este taller se obtendrá una visión compartida del producto que servirá para aceptar y rechazar los requisitos del proyecto.

Esta información resumida podría presentarse en una sola página o en una caja con el tamaño y formato de las utilizadas para cereales.

Ejemplo de caja de visión



Frente de la caja

- ✓ Nombre del proyecto
- ✓ Logotipo y slogan
- ✓ Argumentos principales para comprar el producto

Laterales de la caja

- ✓ Necesidades técnicas
- ✓ Usuarios principales
- ✓ Funcionalidades principales

Parte posterior de la caja

- ✓ Características
- ✓ Funcionalidades avanzadas
- ✓ Requisitos del proyecto

Ejemplo de visión del producto en una hoja

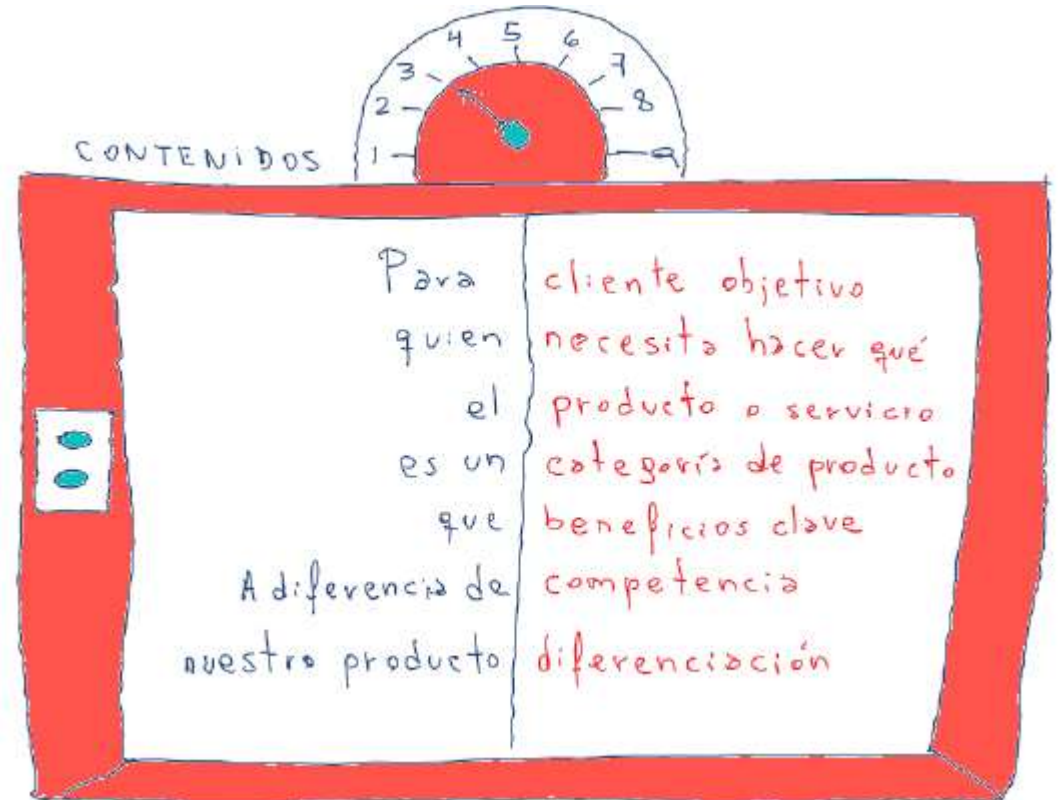
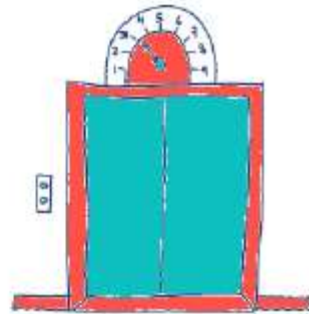
Visión del producto: 			
Grupo objetivo	Necesidad	Producto	Objetivos
 ¿Quiénes serán beneficiados (clientes)? ¿Quién usarán el producto (usuarios)?	 ¿Por qué el cliente o usuario necesita el producto?	 ¿Qué características o funcionalidades son críticas para el cliente? ¿Qué hace que este producto sea diferente a los otros?	 ¿Cómo beneficiará el producto a la empresa que lo crea?

Visión compartida (3)

Discurso del ascensor

El discurso del ascensor (elevator pitch o elevator statement) es una forma de articular la visión del producto para explicarla a una persona externa al proyecto en pocos minutos. Esta herramienta se utiliza para explorar, desarrollar y compartir una visión del producto.

Se denomina "discurso del ascensor" porque debe explicar brevemente la visión del producto a un interesado en el tiempo máximo que demora desde que sube a un ascensor hasta que se baje del mismo.



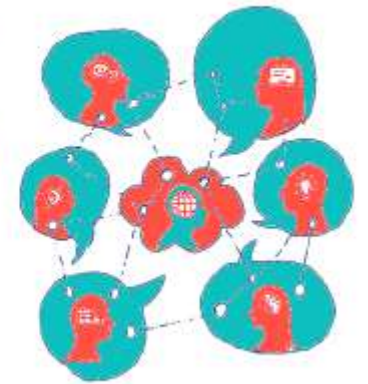
Comunicaciones



Conocimiento compartido

Otro de los conceptos básicos de la filosofía ágil es la de compartir el conocimiento entre los miembros del equipo e interesados clave. De esta forma, si algún miembro del equipo no trabajara más en el proyecto, el resto del equipo puede seguir funcionando con normalidad hasta conseguir un reemplazo.

El conocimiento debe ser compartido entre el equipo, el cliente, la comunidad y la organización.



Radiador de información

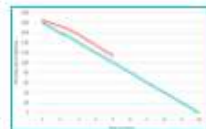
Los radiadores de información son tableros escritos a mano, impresos o electrónicos que un equipo coloca en una ubicación altamente visible, para que todos los miembros del equipo, así como los interesados clave, puedan ver la información más reciente del proyecto de un vistazo. Por ejemplo:



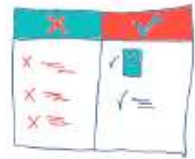
Un **tablero electrónico** de acceso remoto visible en el sitio principal de la organización donde se presenta en tiempo real el número de pruebas automatizadas, la velocidad del equipo, los informes de incidentes, el estado de integración continua, el listado de trabajo pendiente, tendencias, riesgos, etc.



Un **tablero Kanban** con tarjetas y gráficos dibujados a mano que utiliza baja tecnología y alto contacto (low tech – high touch).





Un diagrama de trabajo pendiente (**Burndown chart**) indicando las tareas que faltan completar a través del tiempo.



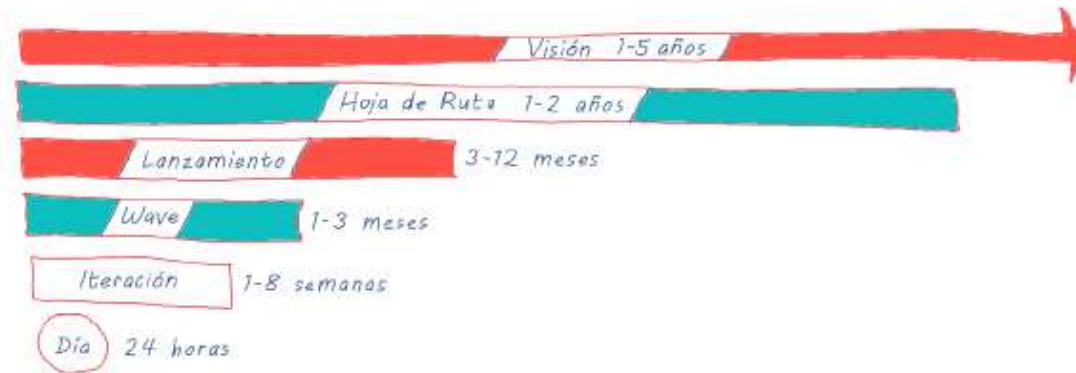
Registros de problemas visualizando los impedimentos que se han resuelto y los que están pendientes de resolución.

Participación de los interesados en las reuniones

Reunión	Tipo de participación de los interesados externos al equipo	
Estimación de historias	Alta 	Fundamental en las discusiones preliminares sobre los requisitos y la definición de las historias de usuario.
Priorización del product backlog	Media 	Mantener una comunicación fluida para que provean información antes que el dueño del producto decida las prioridades.
Planificación de la iteración	Media 	Podrían participar para aportar información al equipo sobre dependencias, riesgos, oportunidades, recursos compartidos, valor, etc.
Diaria	Baja 	Podrían asistir a la reunión, pero sin participar. Cualquier consulta o aporte, la debería realizar al líder ágil después de la reunión diaria.
Revisión de la iteración	Alta 	Fundamental para escuchar su retroalimentación sobre el producto y mantenerlos involucrados para mejorar la calidad del proyecto.
Retrospectiva	Baja 	Podrían participar algunos interesados clave vinculados al proyecto.

Planificación ágil

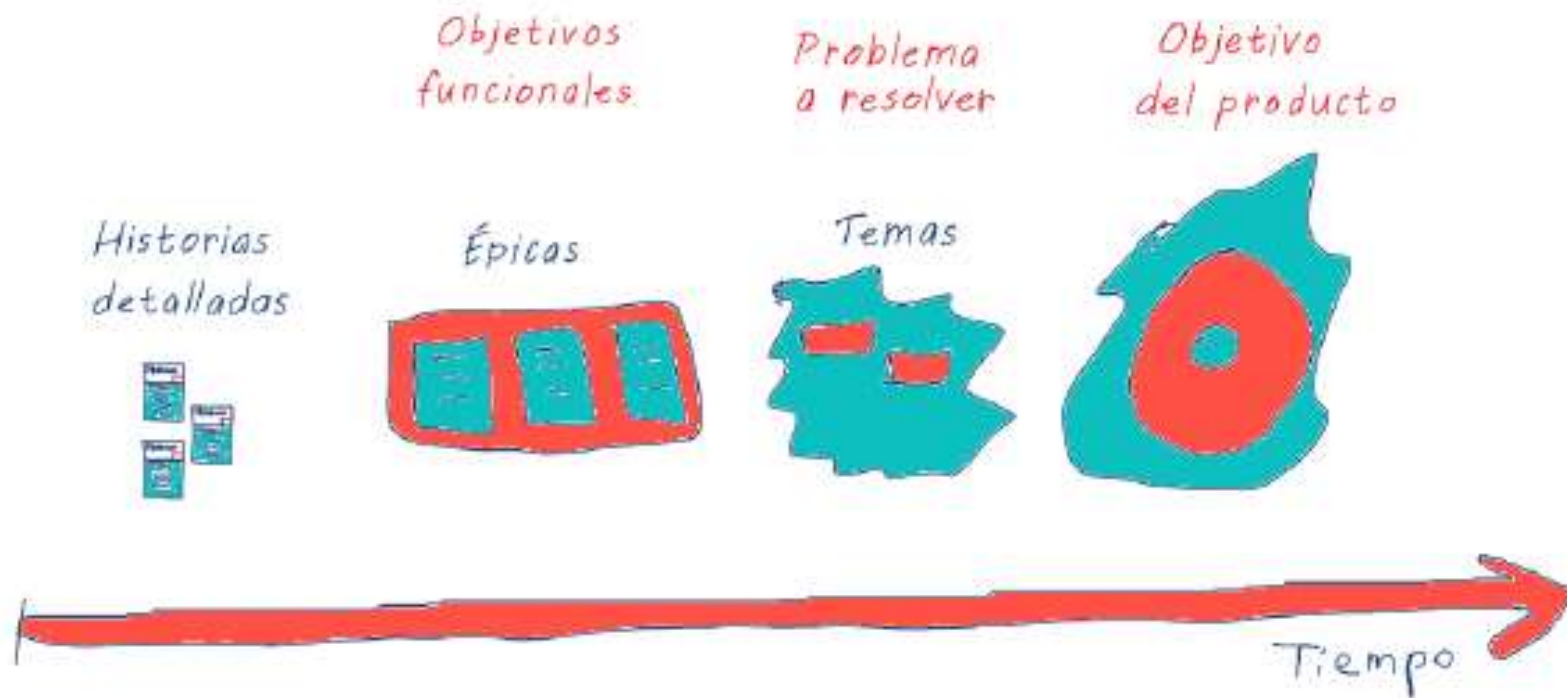
La planificación de proyectos ágiles ocurre en diferentes horizontes temporales.



Nota: Las duraciones del gráfico son solo ejemplos y varían según el tipo de proyecto.

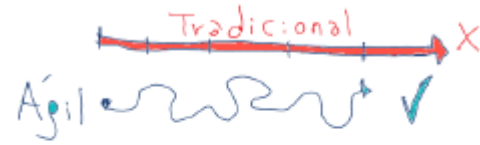
- **Visión del producto:** qué es el producto, quién lo usará y por qué lo usará. Cómo el producto respalda la estrategia de la organización. Revisión anual.
- **Hoja de ruta del producto:** mapa con los requisitos de alto nivel del producto y los plazos para los lanzamientos. Sirve tanto para alinear a los interesados como para adquirir el presupuesto del proyecto. Revisión semestral o cuando lo solicite el cliente.
- **Lanzamiento:** cronograma de alto nivel para los lanzamientos sobre las funcionalidades del producto. Revisión trimestral.
- **Wave:** épicas e historias de usuario detalladas para las próximas iteraciones y a nivel más agregado para lo que se podría realizar más adelante. Revisión mensual.
- **Iteración:** objetivos, tareas y requisitos de la iteración. Revisión al final de cada iteración.
- **Día:** actividades por realizar durante un día de trabajo. Revisión diaria.

Elaboración progresiva

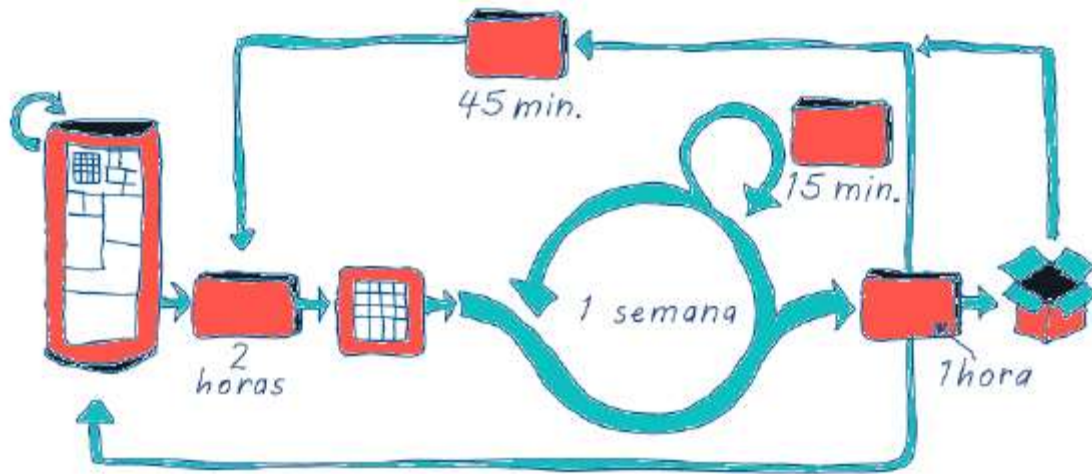


Adaptabilidad

Las metodologías ágiles ponen más énfasis en la adaptabilidad de los planes con relación a la previsibilidad de seguir un plan al pie de la letra.



Los cambios de requisitos son un aspecto natural, inevitable y hasta deseable del desarrollo de proyectos ágiles. Adaptarse a los cambios de requisitos en cualquier momento del proyecto es más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos luego para controlar los cambios en los requisitos.



Timebox

La técnica del timebox (periodo de tiempo preestablecido) consiste en fijar el tiempo máximo de un intervalo de tiempo para conseguir unos objetivos, tomar una decisión o realizar tareas.



Una vez fijado este intervalo de tiempo (ej. iteración de 15 días), el equipo debe hacer lo mejor que pueda en ese plazo. De esta manera, en lugar de que el equipo trabaje en algo hasta completarlo, se acuerda de antemano que tienen un tiempo limitado para desarrollar los entregables.

La aceptación del equipo de esta restricción temporal favorece la priorización de objetivos y tareas ajustando el alcance del producto en función del plazo disponible.

Las iteraciones de tiempo preestablecido en los proyectos ágiles son fundamentales para poder implementar las adaptaciones de manera ordenada, ya que al finalizar cada iteración se utilizan las lecciones aprendidas que podrían cambiar el plan de la próxima iteración.

Adaptación de procesos

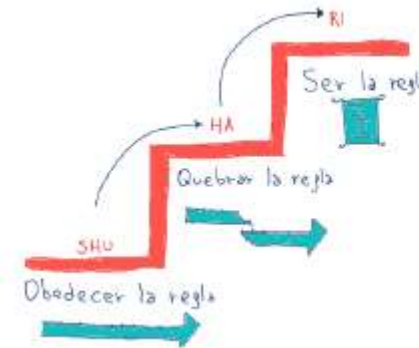
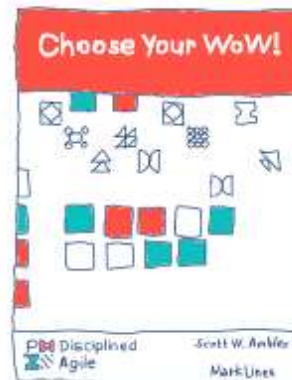
Adaptación de procesos (Tailoring)

Las metodologías ágiles no están pensadas para ser de una "talla única" que aplique a todos los proyectos, sino que los procesos ágiles pueden modificarse según las características de cada proyecto como el tamaño del equipo, contexto, recursos disponibles, criticidad, etc.



Algunos puristas ágiles piensan que el "tailoring" no es recomendable. En la Guía Scrum se menciona: *"Los roles, eventos, artefactos y reglas de Scrum son inmutables y, aunque es posible implementar solo partes de Scrum, el resultado no es Scrum. Scrum existe solo en su totalidad y funciona bien como un contenedor para otras técnicas, metodologías y prácticas"*. Por ejemplo, si un equipo decide eliminar las reuniones de retrospectiva porque creen que no agregan valor, podría ocurrir que la herramienta "Scrum" dejara de funcionar correctamente.

En el otro extremo, Disciplined Agile (DA), a diferencia de otras metodologías que se presentan como un marco metodológico con pasos a seguir, ofrece opciones para adaptar los marcos existentes a cada situación en particular, brindando consejos sobre cuándo y cómo aplicar las diferentes metodologías. DA fue diseñado para ofrecer un equilibrio entre los métodos con un enfoque demasiado estrecho (ej. Scrum) o aquellos demasiado detallados (ej. AUP).



La filosofía **Shu-Ha-Ri** puede ser de ayuda al momento de tomar la decisión si adaptar o no una metodología.

Por último, como parte de la filosofía ágil, las mejoras o adaptación del proceso se pueden plantear en la retrospectiva de cada iteración, para implementar las mejoras en la siguiente iteración.

Mejora continua

Los bucles de retroalimentación son mecanismos que se utilizan para validar y obtener retroalimentación negativa y positiva de manera inmediata sobre el proceso de desarrollo del producto. Esta retroalimentación es la base para la mejora continua.



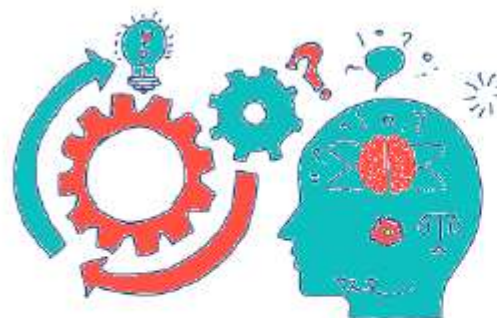
Para el éxito de un proyecto ágil, es fundamental crear una cultura organizacional para implementar pequeñas mejoras a lo largo de todo el proyecto.

El proceso ágil tiene puntos de control integrados para facilitar la retroalimentación y la colaboración mediante reuniones tales como:

1. **Diaria:** permite a los miembros del equipo compartir actualizaciones de estado e identificar los obstáculos que se interponen en su camino.
2. **Revisión de la iteración:** se presenta el incremento del producto al dueño del producto y otros interesados clave para evaluar el proyecto y comentar sobre las necesidades aún no satisfechas.
3. **Retrospectiva:** permite al equipo mirar hacia atrás en lo que salió bien y lo que podría hacerse mejor.



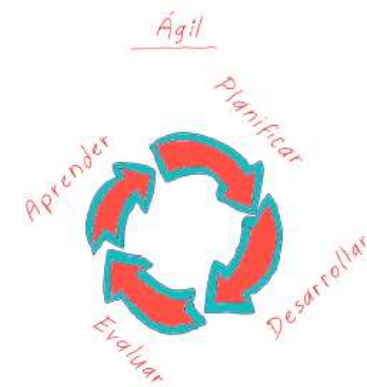
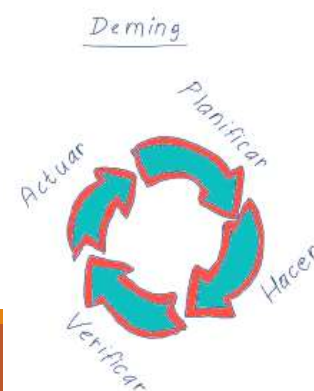
Ciclo de aprendizaje



En el desarrollo de proyectos ágiles se suele comenzar con un bajo conocimiento de cómo será el producto final, por lo que es necesario un aprendizaje permanente para ir mejorando el desarrollo del producto para entregar el máximo valor al cliente.

Este aprendizaje continuo sobre los requisitos del cliente y los cambios tecnológicos son los que justifican mantener ciclos cortos de aprendizaje en cada iteración para poder entregar un producto viable.

Al mantener ciclos de aprendizaje cortos, los nuevos conocimientos que se adquieren en cada iteración se pueden incorporar rápidamente al proyecto.



Mejorar el producto (1)

El desarrollo iterativo e incremental con una retroalimentación constante del cliente es una forma para la mejora continua del producto.



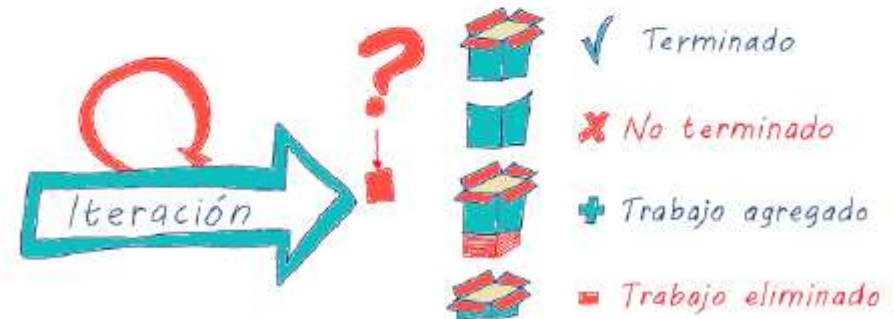
Mediante este ciclo de desarrollar pequeños incrementos, revisarlos, discutir como mejorarlos e implementar pequeñas mejoras, el producto o servicio se va desarrollando a través del proceso de mejor continua.

Revisión de la iteración (Sprint review)

La revisión de la iteración es una reunión que se realiza al final de cada iteración, donde el equipo de desarrollo presenta al dueño del producto el incremento completado para obtener comentarios sobre el producto desarrollado.

Durante esta reunión, el dueño del producto, quien representa los intereses del cliente, revisa si el trabajo realizado cumple con el objetivo de la iteración y con la definición de terminado.

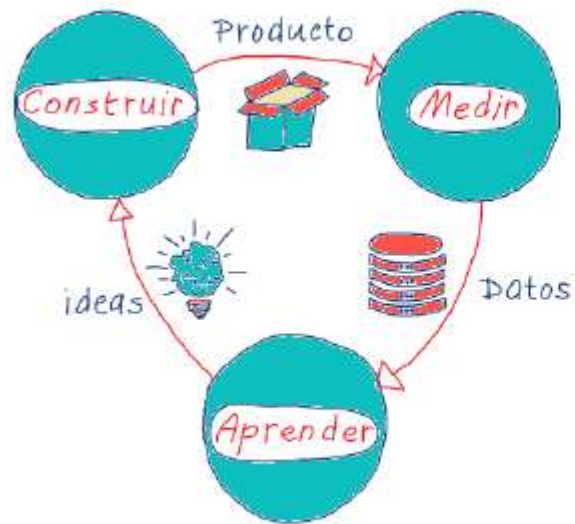
Algunos clientes solicitan una demostración del producto para probarlo, antes de aceptar si cumple o no con la definición de terminado. Como resultado de esta revisión, el dueño del producto determina el estado del trabajo implementado en la iteración:



El resultado final de esta reunión es una revisión del product backlog para priorizar y planificar las actividades de la próxima iteración. También se actualiza el plan de lanzamiento cuando sea necesario.

Mejorar el producto (2)

Ciclo de retroalimentación del producto (Product feedback loop)



La actividad fundamental de una startup es convertir ideas en productos, medir cómo responden los clientes y luego aprender si pivotar o perseverar. Esto sirve como base para el marco "construir-medir-aprender".³⁰

Construir una idea en algo tangible que los clientes deberían amar.

Medir si a los clientes les encanta el producto.

Aprender de los errores para crear un producto que a los clientes les encantará.

El ciclo de retroalimentación del producto es el proceso de recopilar continuamente los comentarios de los clientes y mejorar el producto en función de sus opiniones.

Demostración del sistema (system demo)

En la metodología SAFe, cuando hay varios equipos ágiles desarrollando diferentes componentes de un producto incremental, al terminar la iteración se lleva a cabo una demostración del sistema para presentar a los interesados todos esos componentes integrados.



La mejor medida real de valor, velocidad y progreso del proyecto es la demostración del trabajo totalmente integrado de todos los equipos.

La demostración del sistema es presentada en conjunto por los gerentes del producto y dueños del producto de cada equipo. Los principales interesados de esta demostración son los dueños del negocio, patrocinadores, clientes y otros interesados clave.

Cada demostración brinda a los interesados una medida objetiva del progreso de desarrollo del producto para que puedan proveer una retroalimentación inmediata.

El objetivo de la demostración del sistema es aprender de la experiencia de desarrollo más reciente para ajustar el curso de acción de las próximas iteraciones.

Mejorar el equipo

La mejora continua de los miembros del equipo será fundamental para poder mejorar los procesos y el producto.

Las metodologías ágiles incluyen momentos de reflexión a lo largo del proyecto para evaluar el rendimiento de las personas con el objetivo de mejorar su desarrollo individual y colectivo.



La motivación permanente del equipo es clave para la mejora continua. Algunos factores para aumentar la motivación de un equipo ágil suelen ser:



- Reducir el tamaño del equipo tanto como sea posible
- Empoderar al equipo para que ellos tomen sus propias decisiones
- Conectar al equipo con los usuarios
- Permitir que el equipo sea dueño de su propio proceso
- Brindar al equipo una porción del sistema donde puedan ser dueños de la definición, entrega y despliegue de valor

Las reuniones de retrospectiva son un evento muy importante para que el equipo pueda compartir el conocimiento y pueda realizar auto evaluaciones de manera frecuente.

Reuniones de retrospectivas

El objetivo de esta reunión es mejorar la productividad, las habilidades del equipo y la calidad del producto. En este evento participa todo el equipo y se realiza después de la reunión de revisión de la iteración.

Las reuniones de retrospectiva permiten al equipo mirar hacia atrás para reflexionar en lo que salió bien y lo que podría hacerse mejor para seguir mejorando en las próximas iteraciones.

Como resultado de la reunión de retrospectiva se obtiene:

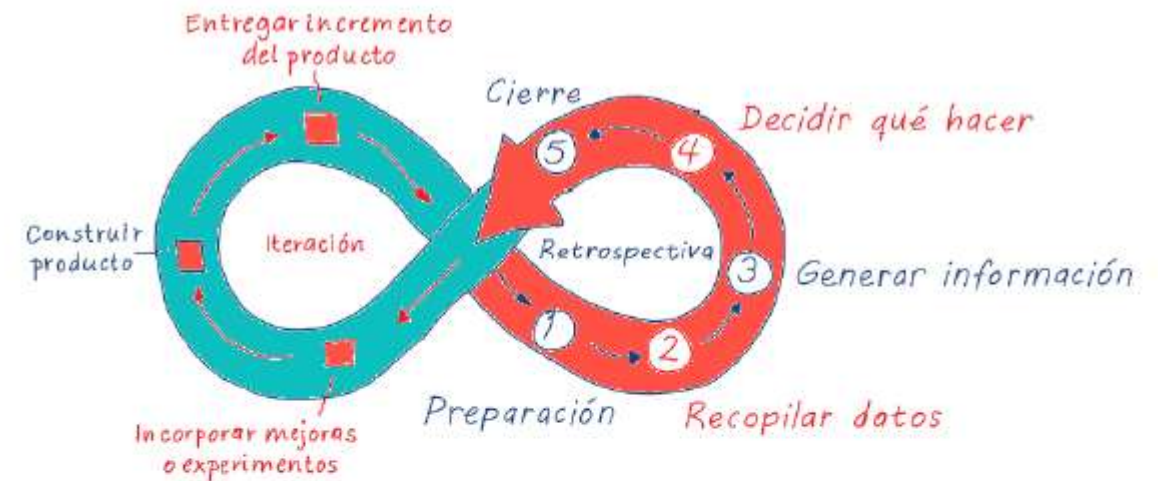
- ✓ Plan de mejora del proceso
- ✓ Mejora en la calidad del producto
- ✓ Mejora en la productividad del equipo

Además de las retrospectivas de la iteración que es la más común, también pueden realizarse retrospectivas más largas e intensivas en hitos importantes del proyecto. Por ejemplo:

- Retrospectivas de lanzamiento
- Retrospectivas sorpresa (realizadas cuando un evento inesperado cambia la situación)
- Retrospectivas de fin del proyecto

Estas retrospectivas más intensivas, brindan la oportunidad de reflexionar más profundamente sobre las experiencias y condensar lecciones clave para compartir con el resto de la organización.

Si el equipo quisiera aumentar la tasa de mejora de procesos se podrían llevar a cabo retrospectivas más seguido para identificar mejoras.



- ① **Preparación** / El coach ágil en conjunto con el equipo, definen las reglas básicas para garantizar que todos se sientan cómodos discutiendo los problemas del proyecto.
- ② **Recopilar datos** / El equipo recopila datos de los problemas que se enfrentan durante la iteración mediante diagrama de espina de pescado, 5 porqués, gráficos de control, etc.
- ③ **Generar información** / Basándose en los datos recopilados, el equipo puede inferir la causa raíz de los problemas que se enfrentan, transformando esos datos en información.
- ④ **Decidir qué hacer** / Basándose en la información, el equipo decide los planes de mejora que se implementarán en las próximas iteraciones para no cometer los mismos problemas.