CORFO
BecasCapital Humano

dalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 7

FUNDAMENTOS DE DESARROLLO AGILE

Manual Módulo 7: Fundamentos de desarrollo Agile

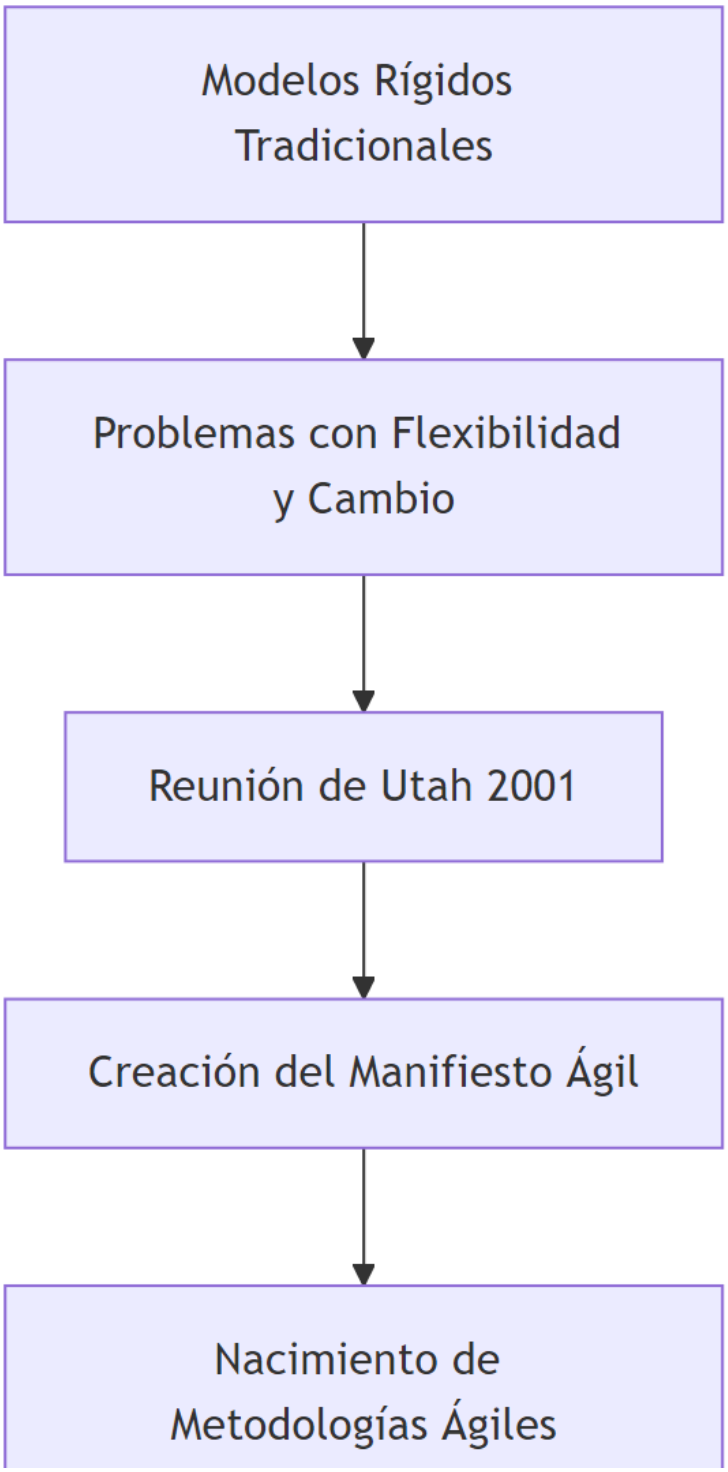
Introducción

En este módulo, explorarán los fundamentos de las **metodologías ágiles**, que han revolucionado la forma en que se desarrollan los proyectos de software. Las metodologías ágiles permiten a los equipos trabajar de manera más eficiente y flexible, respondiendo rápidamente a los cambios en los requisitos y prioridades. A través de prácticas interactivas y colaborativas, Agile promueve la entrega continua de valor al cliente. Durante el módulo, aprenderás sobre el **Manifiesto Ágil**, que establece los valores y principios clave, y conocerás las principales metodologías ágiles, como **Scrum**, **Kanban**, y **Extreme Programming (XP)**. También veremos cómo estas metodologías difieren de los enfoques tradicionales, como **Waterfall** y **RUP**, y cómo pueden aplicarse para mejorar el trabajo en equipo y la calidad del producto. Al finalizar, tendrás una comprensión clara de los beneficios de adoptar Agile en proyectos de desarrollo y cómo implementarlo de manera efectiva en tu equipo.

1. Metodologías Ágiles

1.1 Orígenes de las metodologías ágiles

Las **metodologías ágiles** surgieron como una respuesta a las limitaciones de los enfoques tradicionales de desarrollo de software, como **Waterfall**. Durante los años 90, se observó que los modelos rígidos y secuenciales no eran adecuados para proyectos en los que los requisitos cambiaban con frecuencia. En 2001, 17 desarrolladores se reunieron en Utah para crear el **Manifiesto Ágil**, que marcó el inicio formal de Agile como metodología de desarrollo. Este manifiesto busca mejorar la entrega de software al enfocarse en la colaboración, flexibilidad y entrega continua de valor.


Modelos Rígidos Tradicionales
Problemas con Flexibilidad y Cambio
Reunión de Utah 2001
Creación del Manifiesto Ágil
Nacimiento de Metodologías Ágiles

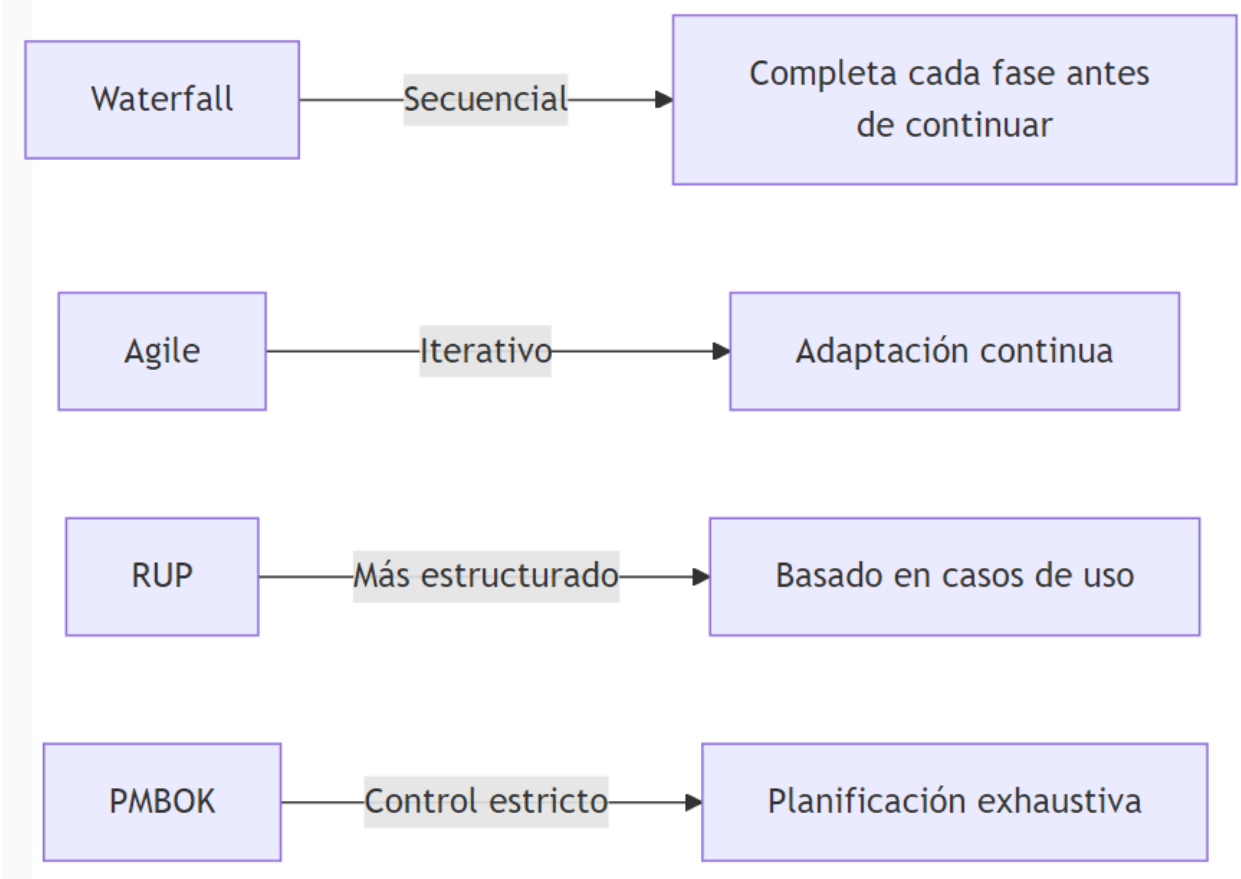
1.2 Diferencias con modelos tradicionales de desarrollo (Waterfall, RUP, Modelo PMBOK)

Las metodologías ágiles difieren notablemente de los enfoques tradicionales en varios aspectos. A continuación, se destacan las diferencias clave con tres enfoques tradicionales:

- **Waterfall**: Un modelo secuencial donde cada fase debe completarse antes de avanzar. Los cambios son difíciles de manejar una vez que se pasa de fase.

- **RUP (Rational Unified Process)**: Aunque es iterativo, RUP sigue un enfoque más estructurado y dirigido por casos de uso, mientras que **Agile** es más flexible y adaptativo a los cambios.

- **PMBOK**: Un enfoque que prioriza la planificación detallada y el control estricto del proyecto. Agile, en cambio, promueve una mayor flexibilidad y adaptación continua a los requisitos.



1.3 Diferencias entre “Agile”, “Agilidad” y “Agilismo”

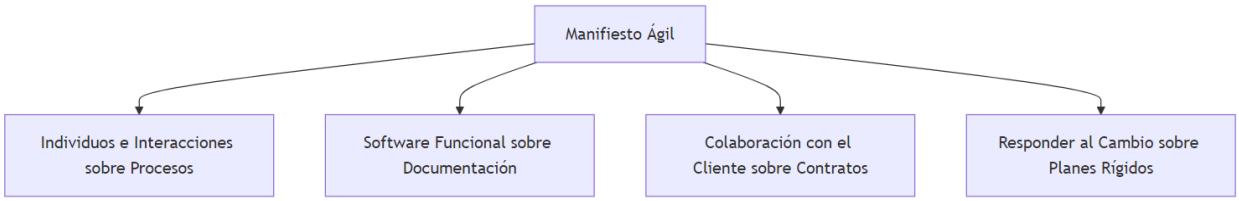
- **Agile**: Es el conjunto de metodologías que implementan los principios del **Manifiesto Ágil** (Scrum, XP, Kanban).
- **Agilidad**: Se refiere a la capacidad de una organización o equipo para adaptarse rápidamente a los cambios y ajustarse a las nuevas demandas.
- **Agilismo**: Describe la cultura o mentalidad de una organización que sigue los principios de **Agile** en todos sus aspectos, no solo en el desarrollo de software.

1.4 Manifiesto Ágil

El **Manifiesto Ágil** fue creado en 2001 por un grupo de desarrolladores con el objetivo de mejorar el proceso de desarrollo de software. El manifiesto se basa en cuatro valores fundamentales:

1. **Individuos e interacciones** sobre procesos y herramientas.
2. **Software funcional** sobre documentación extensiva.
3. **Colaboración con el cliente** sobre negociación contractual.
4. **Responder al cambio** sobre seguir un plan rígido.

Además, se establecieron 12 principios que promueven la entrega continua de software funcional, la adaptación a cambios, y la colaboración con el cliente.

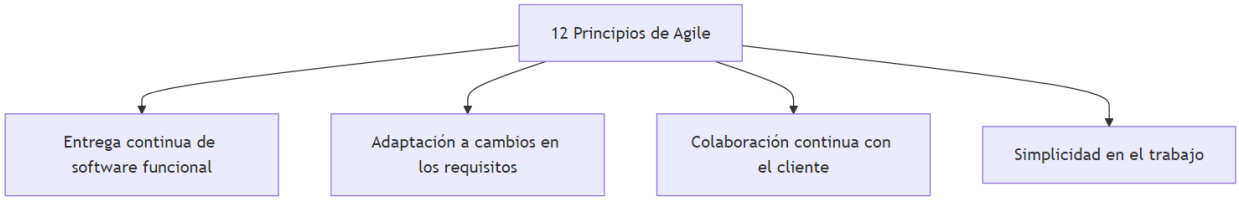


1.5 Valores y Principios del Manifiesto Ágil

El Manifiesto Ágil se complementa con 12 principios que ayudan a guiar la implementación de **Agile** en proyectos de desarrollo. Algunos de los principios clave incluyen:

- **Entrega continua**: Se busca entregar software funcional de manera frecuente, idealmente cada pocas semanas.
- **Adaptación a los cambios**: Se aceptan cambios en los requisitos incluso en fases avanzadas.

- **Colaboración**: Se promueve la comunicación constante entre el equipo de desarrollo y el cliente.
- **Simplicidad**: Se fomenta la eliminación de trabajo innecesario.

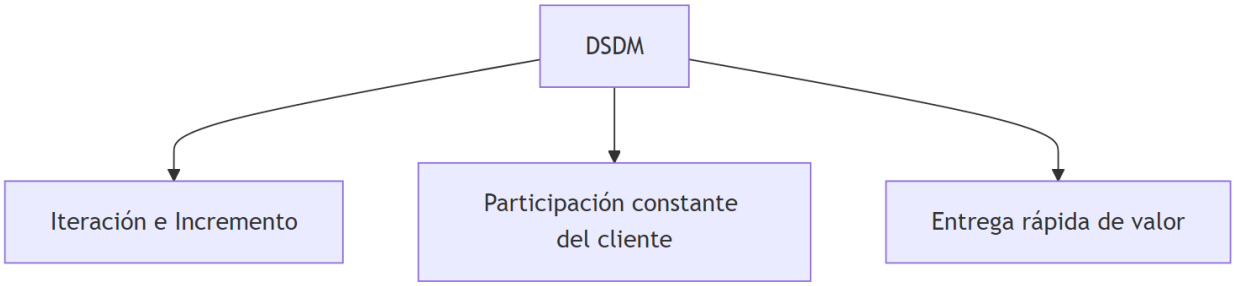


1.6 Abanico de Metodologías Ágiles

Existen diferentes metodologías que implementan los principios ágiles de distintas maneras. A continuación, se destacan algunas de las más populares:

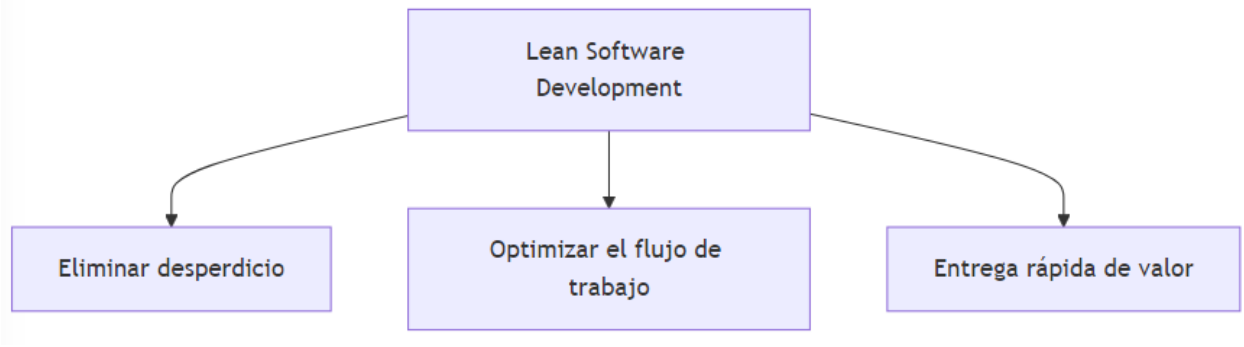
1.6.1 DSDM-Atern

DSDM (Dynamic Systems Development Method) es una metodología ágil que se centra en la entrega rápida y colaborativa de proyectos. Promueve el desarrollo iterativo e incremental, con participación constante del cliente para asegurar que el producto final cumple con sus expectativas y necesidades.



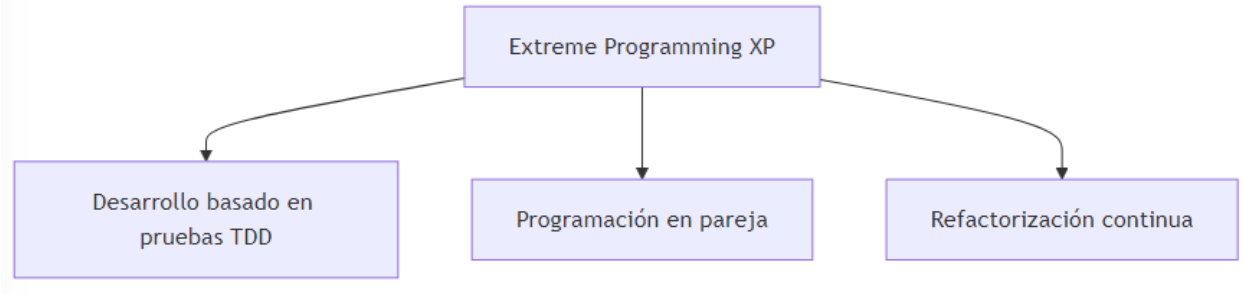
1.6.2 Lean Software Development

El **Lean Software Development** proviene del enfoque **Lean Manufacturing** y se centra en optimizar el flujo de trabajo, eliminando todo lo que no aporta valor. Se enfoca en reducir el desperdicio y entregar el producto lo más rápido posible al cliente.



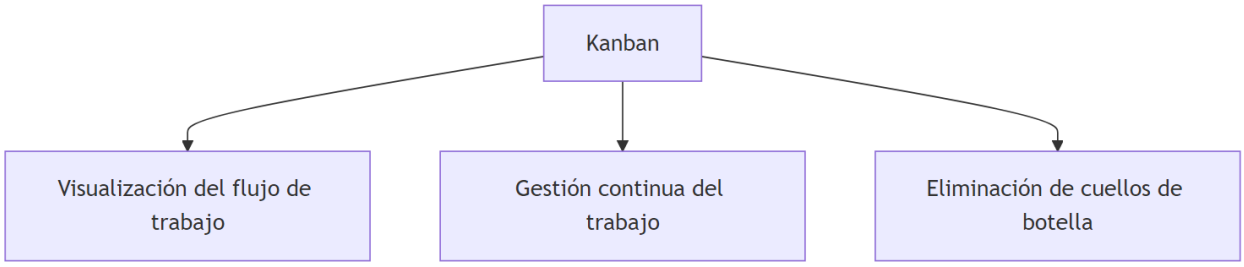
1.6.3 Extreme Programming (XP)

Extreme Programming (XP) es una metodología ágil que se centra en mejorar la calidad del software mediante prácticas técnicas como la **programación en pareja**, el **desarrollo basado en pruebas (TDD)**, y la **refactorización continua**. XP fomenta una alta colaboración entre el equipo de desarrollo y el cliente.



1.6.4 Kanban

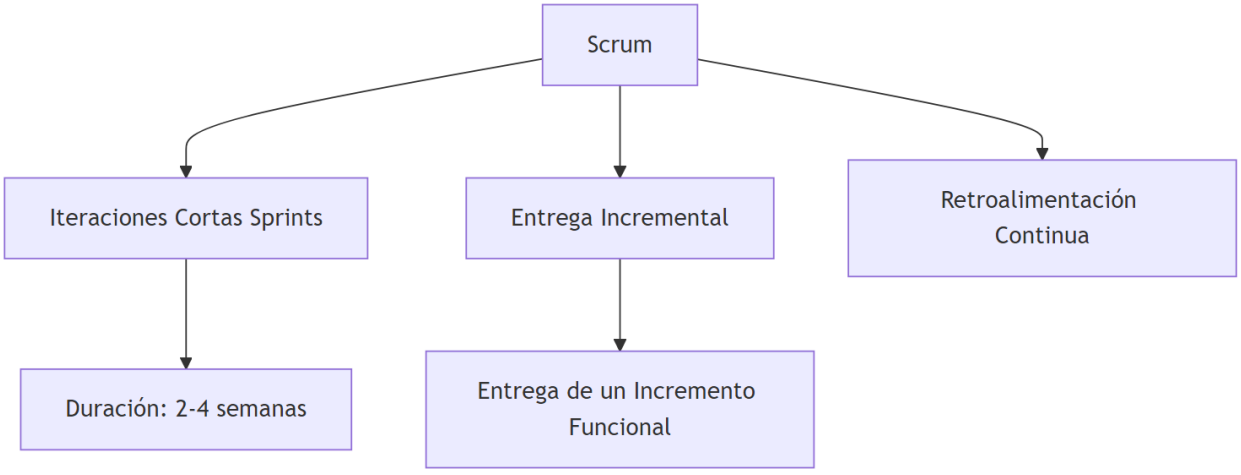
Kanban es un enfoque visual para gestionar el flujo de trabajo en un proyecto. Utiliza un **tablero Kanban** que muestra las tareas en diferentes fases de progreso, como "Pendiente", "En Progreso" y "Hecho". Esta metodología permite una gestión continua del trabajo en curso y facilita la identificación de cuellos de botella.



2. Fundamentos de Scrum

2.1 Qué es Scrum

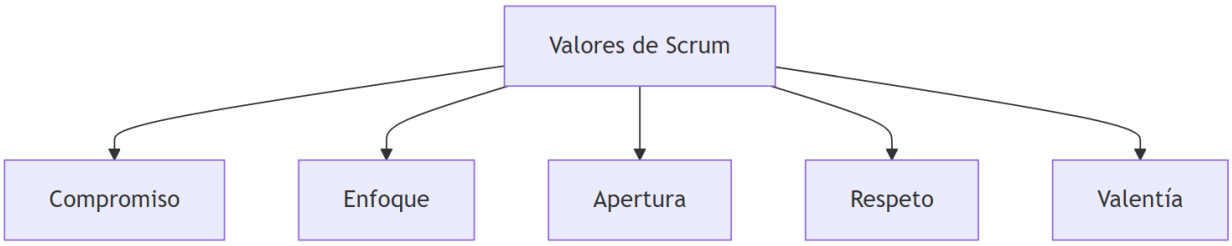
Scrum es un marco de trabajo ágil que permite a los equipos desarrollar, entregar y mantener productos de manera eficiente mediante iteraciones cortas y ciclos de retroalimentación continua. Es ampliamente utilizado en proyectos de desarrollo de software, pero también puede aplicarse a otros contextos. Scrum se basa en la entrega incremental de valor a través de **sprints** que suelen durar entre 2 y 4 semanas, durante los cuales se completa un conjunto de tareas seleccionadas del **Product Backlog**.



2.2 Principios y Valores de Scrum

Los valores y principios de **Scrum** se alinean con los fundamentos del **Manifiesto Ágil** y se centran en la colaboración, transparencia y mejora continua. Los cinco valores clave de Scrum son:

1. **Compromiso**: El equipo se compromete a alcanzar los objetivos del sprint.
2. **Enfoque**: Los miembros del equipo se concentran en el trabajo del sprint y los objetivos del mismo.
3. **Apertura**: El equipo mantiene una comunicación transparente sobre los desafíos y el progreso.
4. **Respeto**: Los miembros del equipo se respetan mutuamente como profesionales.
5. **Valentía**: El equipo tiene la valentía de tomar decisiones difíciles y comprometerse a entregar valor.

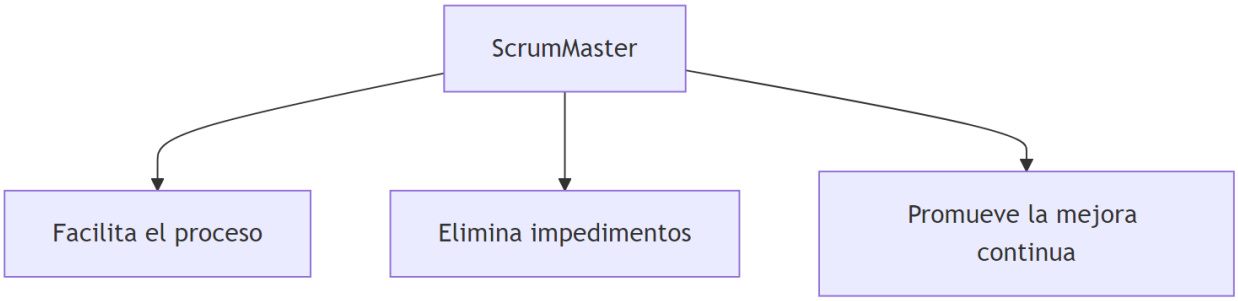


2.3 Roles en Scrum

En **Scrum**, hay tres roles fundamentales que aseguran el éxito del equipo y la correcta implementación del proceso ágil:

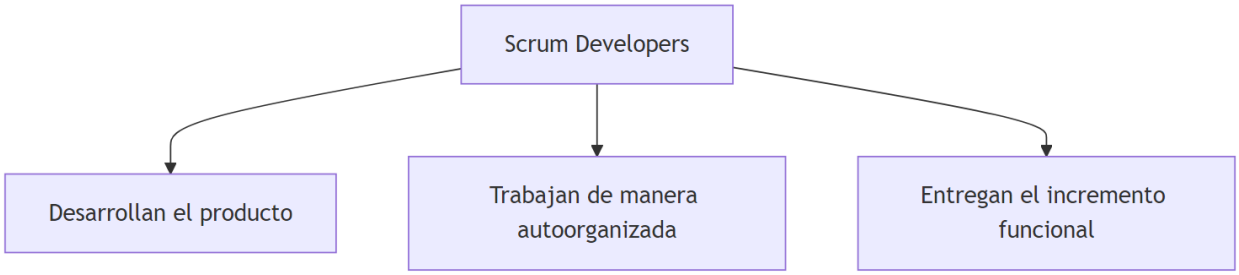
2.3.1 ScrumMaster

El **ScrumMaster** es responsable de facilitar el proceso Scrum y asegurarse de que el equipo siga los principios y prácticas ágiles. Actúa como un coach, ayudando al equipo a eliminar impedimentos, promoviendo la mejora continua y asegurando que los principios de Scrum se mantengan.



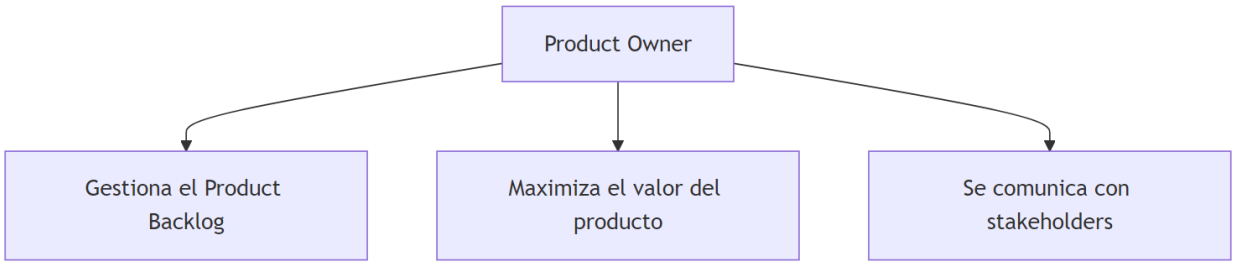
2.3.2 Scrum Developers

Los **Scrum Developers** son los miembros del equipo responsables de realizar el trabajo técnico. Ellos se encargan de desarrollar, probar y entregar el incremento funcional del producto al final de cada sprint. Los developers trabajan de manera colaborativa, autoorganizada y multidisciplinar.



2.3.3 Product Owner

El **Product Owner** es el responsable de maximizar el valor del producto que el equipo está desarrollando. Administra el **Product Backlog** y se asegura de que las tareas más importantes y que aportan mayor valor al producto se prioricen. También es el punto de contacto principal con los stakeholders.

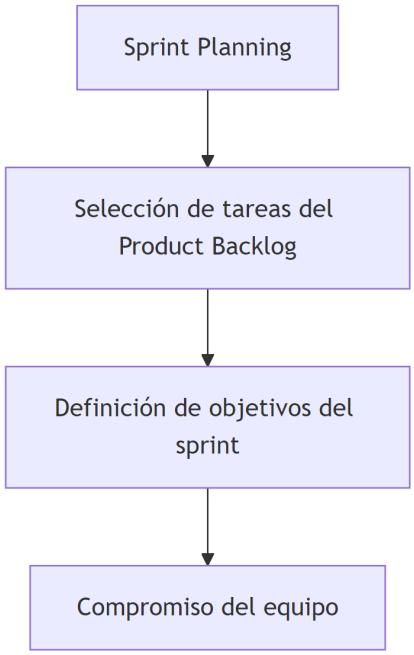


2.4 Prácticas en Scrum

Las prácticas en **Scrum** incluyen ceremonias específicas que ayudan a organizar el trabajo, mantener la transparencia y garantizar la mejora continua.

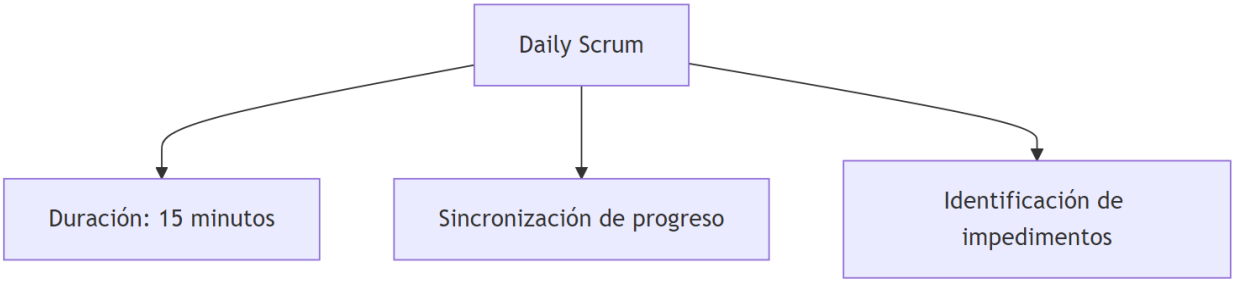
2.4.1 Sprint Planning

La **Sprint Planning** es la reunión en la que el equipo planifica el trabajo que se realizará durante el sprint. El **Product Owner** presenta las prioridades del **Product Backlog** y el equipo selecciona los elementos que se comprometen a completar en el sprint.



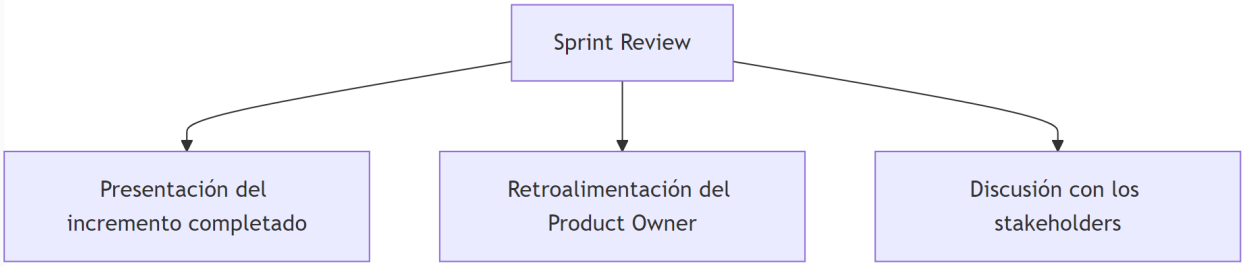
2.4.2 Daily Scrum

El **Daily Scrum** es una reunión diaria de no más de 15 minutos en la que el equipo sincroniza su progreso y discute posibles impedimentos. Es una reunión rápida y enfocada en mantener a todos alineados con los objetivos del sprint.



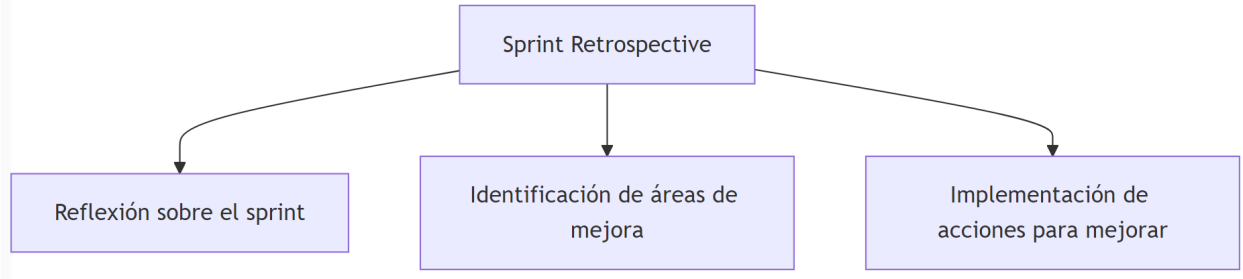
2.4.3 Sprint Review

En la **Sprint Review**, el equipo presenta el incremento completado a los stakeholders y al **Product Owner**. Se discute el progreso del sprint, lo que se completó y lo que no, y se recibe retroalimentación sobre el producto.



2.4.4 Sprint Retrospective

La **Sprint Retrospective** es una ceremonia que se realiza al final del sprint, donde el equipo reflexiona sobre lo que funcionó bien y qué se puede mejorar. El objetivo es implementar mejoras continuas en el proceso para los siguientes sprints.

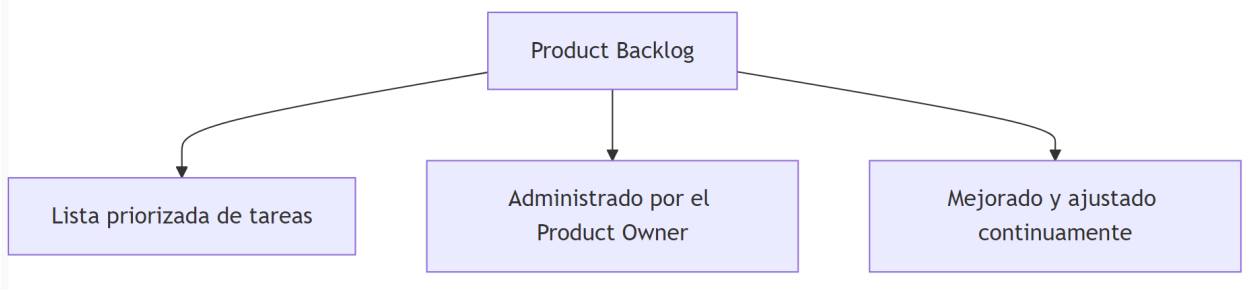


2.5 Artefactos en Scrum

Scrum cuenta con tres artefactos principales que ayudan a organizar y visualizar el trabajo del equipo.

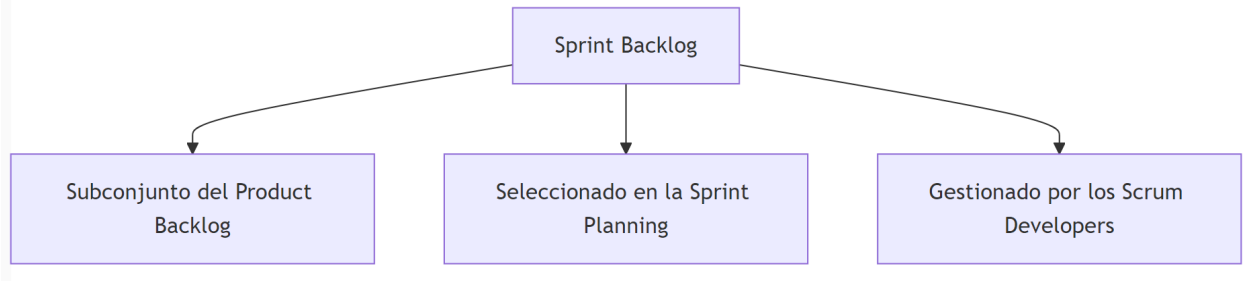
2.5.1 Product Backlog

El **Product Backlog** es una lista priorizada de todas las características, funciones, mejoras y correcciones que se necesitan en el producto. El **Product Owner** es el encargado de mantener y priorizar este backlog para maximizar el valor del producto.



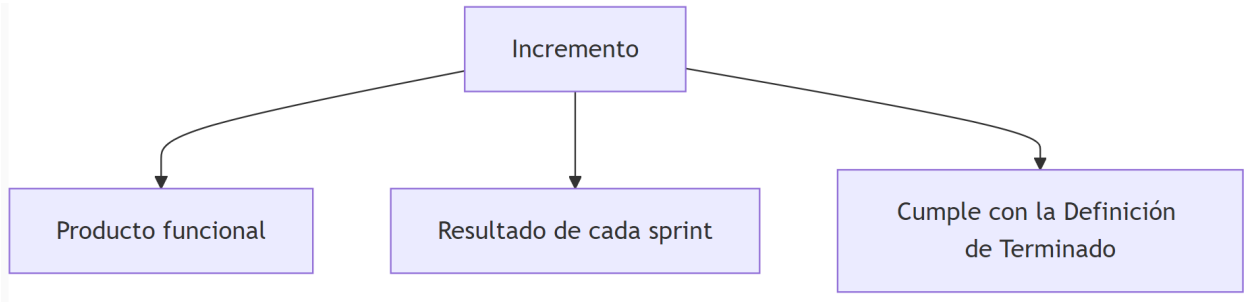
2.5.2 Sprint Backlog

El **Sprint Backlog** es un subconjunto del **Product Backlog** que contiene los elementos que el equipo ha seleccionado para completar durante un sprint. Este backlog es gestionado y actualizado por los **Scrum Developers** durante el sprint.



2.5.3 Incremento

El **Incremento** es el resultado de cada sprint. Es la suma de todos los elementos completados durante el sprint y debe ser un producto funcional y potencialmente desplegable. Cada incremento debe ser de alta calidad y cumplir con la **Definición de Terminado (Definition of Done)** establecida por el equipo.



Este conjunto de roles, prácticas y artefactos en **Scrum** ayuda a los equipos a gestionar de manera eficiente el desarrollo del producto, manteniendo una entrega continua de valor y fomentando la colaboración y la mejora constante.

3. Visión del Producto en Scrum

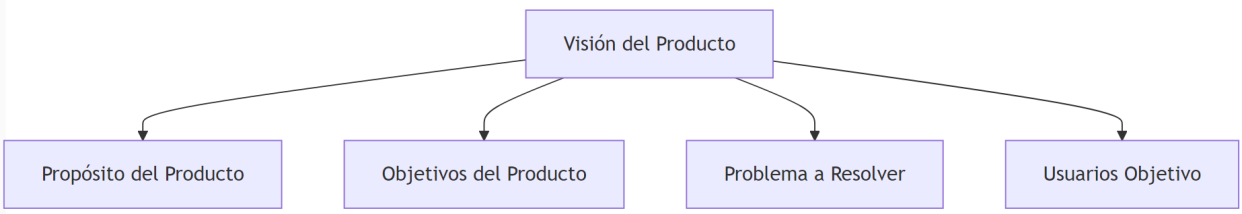
3.1 Producto y su Visión

En **Scrum**, la **visión del producto** es una declaración clara y concisa que describe el propósito, las metas y la dirección del producto. Sirve como una guía que alinea a todo el equipo y a los stakeholders en torno a lo que se quiere lograr con el desarrollo del producto.

Esta visión es responsabilidad del **Product Owner**, quien la comunica a todo el equipo para asegurar que todos trabajen hacia el mismo objetivo.

La visión debe responder a preguntas como:

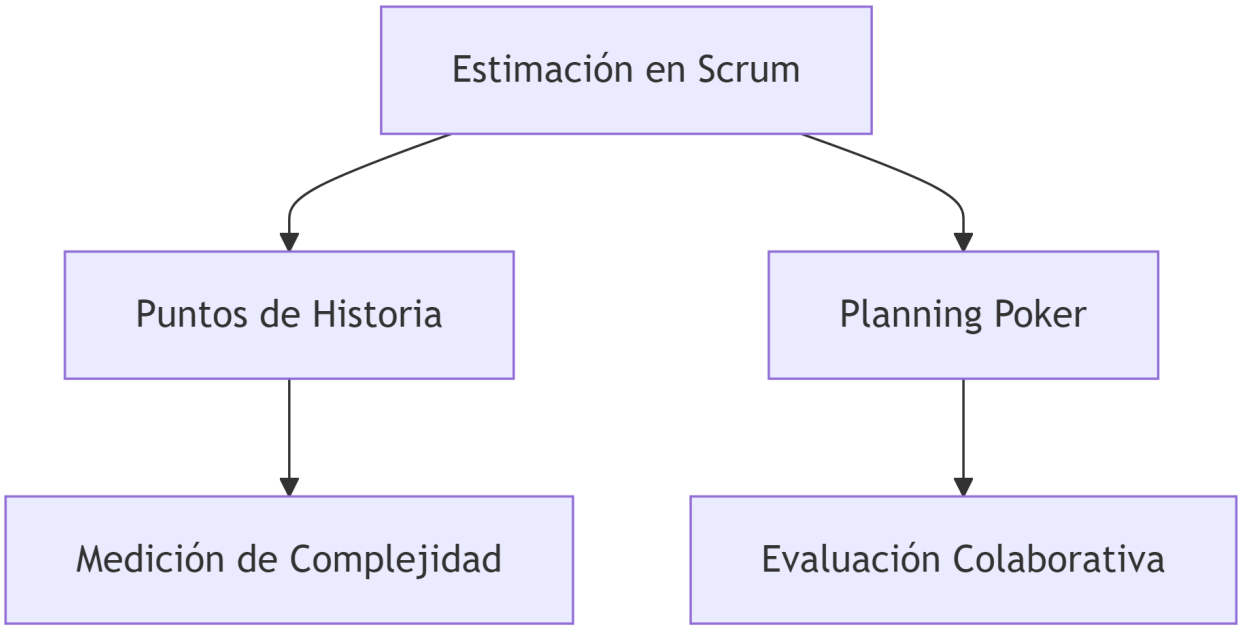
- ¿Cuál es el valor principal que el producto va a ofrecer?
- ¿Quiénes serán los usuarios del producto?
- ¿Cuál es el problema que el producto resolverá?



3.2 Cómo Estimar

La **estimación** en Scrum es el proceso mediante el cual el equipo de desarrollo evalúa la complejidad y el esfuerzo necesario para completar cada tarea o funcionalidad del **Product Backlog**. La estimación generalmente se realiza utilizando "puntos de historia" o técnicas como **Planning Poker**, donde los miembros del equipo asignan un valor numérico (basado en la complejidad) a cada ítem del backlog.

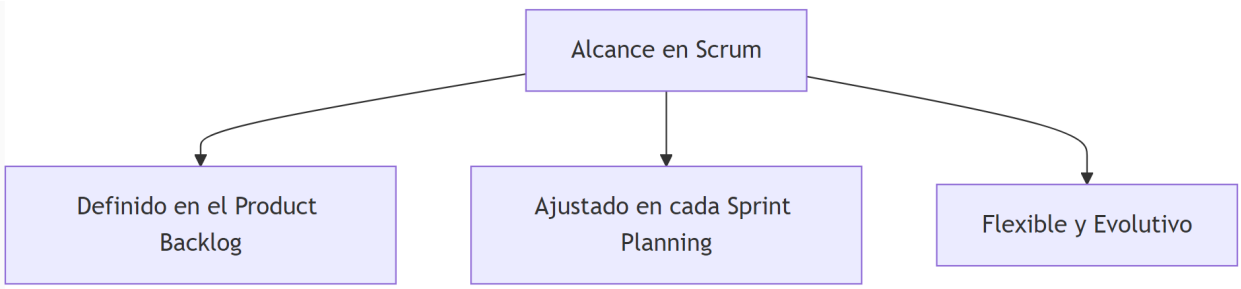
Los puntos de historia no están relacionados con horas de trabajo, sino con la dificultad relativa entre tareas. Esta metodología ayuda al equipo a proyectar cuánto trabajo pueden abordar en un **sprint**.



3.3 Definición de Alcance

El **alcance** en Scrum es flexible y evolutivo. Se define inicialmente en términos generales a través del **Product Backlog**, que contiene todas las funcionalidades deseadas para el producto. Sin embargo, el alcance específico para cada sprint se define durante la **Sprint Planning**, donde el equipo selecciona los elementos que pueden completar durante ese ciclo.

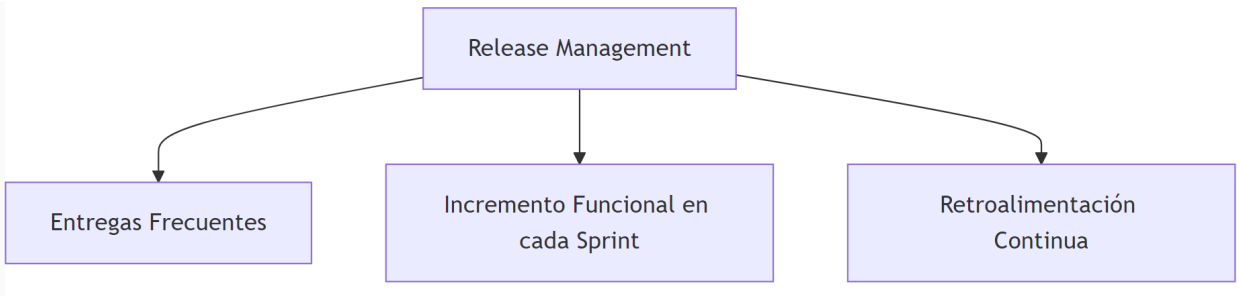
Esto permite que el alcance del proyecto se ajuste en función de la retroalimentación y los cambios en las prioridades. El equipo solo se compromete con el trabajo que se puede realizar dentro del sprint, manteniendo flexibilidad en el resto del desarrollo.



3.4 Release Management

El **Release Management** en Scrum se refiere a cómo se gestionan las entregas del producto o los incrementos funcionales a los usuarios finales. En lugar de una única gran entrega al final del proyecto, Scrum permite **entregas continuas** o **iterativas** después de cada sprint, siempre que el incremento del producto esté listo para su despliegue.

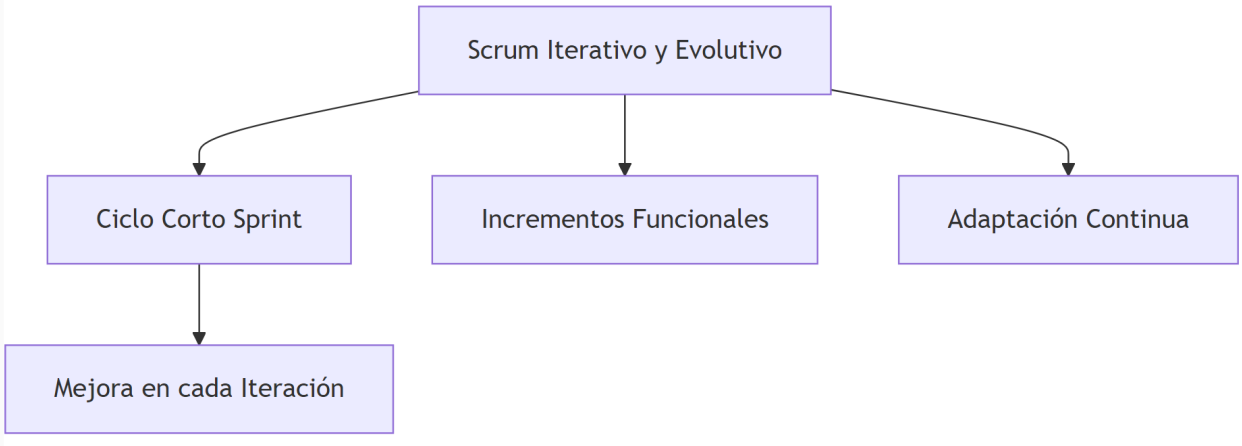
Las releases en Scrum suelen ser más frecuentes, lo que permite obtener retroalimentación del cliente de forma continua y ajustar el desarrollo en función de esta retroalimentación. Esto también reduce los riesgos al ofrecer valor desde las primeras fases del proyecto.



3.5 Scrum como modelo iterativo y evolutivo

Scrum sigue un **modelo iterativo y evolutivo**, donde el desarrollo se realiza en ciclos cortos (sprints). Cada sprint produce un incremento funcional del producto que puede ser desplegado o revisado. Esto permite que el producto evolucione a lo largo del tiempo, con cada iteración refinando las características existentes o agregando nuevas funcionalidades.

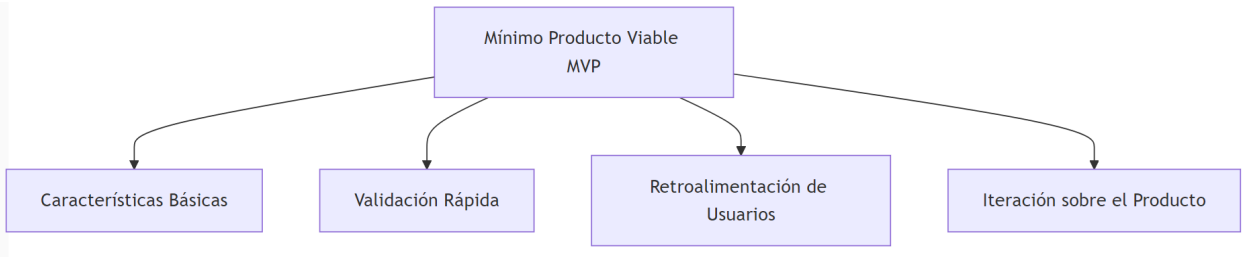
Este enfoque iterativo permite la **adaptación continua** a los cambios en los requisitos o a los nuevos descubrimientos que surgen durante el proceso de desarrollo.



3.6 Mínimo Producto Viable (MVP)

El **Mínimo Producto Viable (MVP)** es una versión del producto que contiene las características mínimas necesarias para que pueda ser lanzado y obtener retroalimentación de los usuarios reales. El objetivo del **MVP** es validar hipótesis de negocio con la menor inversión posible y aprender rápidamente lo que los usuarios necesitan.

En Scrum, el **MVP** puede entregarse en los primeros sprints, permitiendo que los desarrolladores ajusten el producto en función de la retroalimentación obtenida del mercado antes de agregar funcionalidades adicionales.



Este enfoque asegura que el equipo pueda entregar valor de manera temprana y adaptarse según las necesidades reales de los usuarios, reduciendo el riesgo y optimizando el desarrollo.

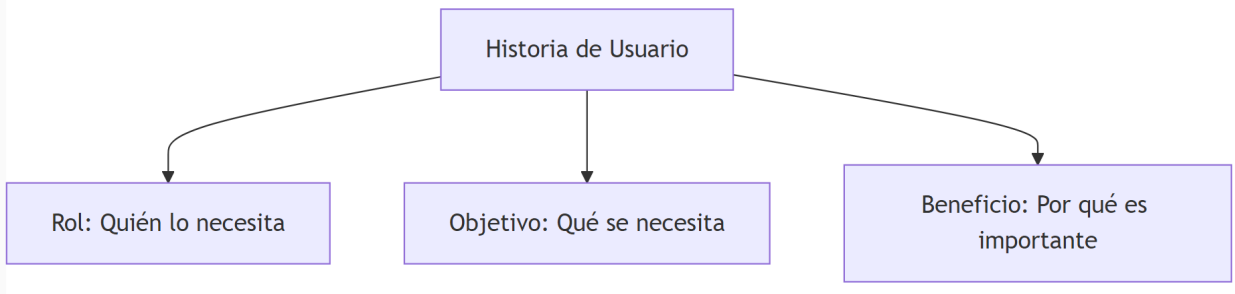
4. Taller de Historias de Usuario

4.1 Qué es una Historia de Usuario

Una **Historia de Usuario** es una breve descripción de una funcionalidad del sistema, escrita desde el punto de vista del usuario final. La estructura típica de una historia de usuario es:

"Como [rol], quiero [objetivo] para [beneficio]"

Este formato captura quién necesita la funcionalidad (rol), qué se debe lograr (objetivo) y por qué es importante (beneficio). Las historias de usuario son fundamentales en **Scrum** para describir el trabajo que se debe hacer en términos comprensibles tanto para los desarrolladores como para los stakeholders.



Ejemplo de una historia de usuario:

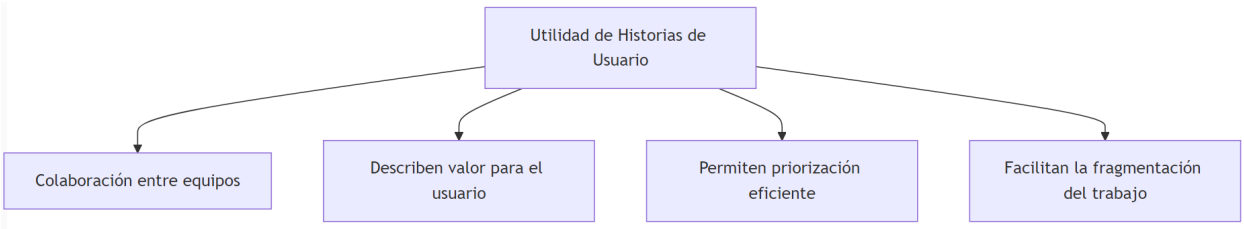
- **Como** usuario registrado, **quiero** recibir notificaciones de mis pedidos, **para** estar informado sobre su estado.

4.2 Utilidad de las Historias de Usuario

Las **historias de usuario** son útiles porque:

- **Facilitan la colaboración**: Al estar escritas en un lenguaje simple, permiten que todos los miembros del equipo (desarrolladores, testers, stakeholders) las comprendan y colaboren en su refinamiento.
- **Describen el valor**: Se enfocan en el valor que las funcionalidades aportan al usuario final, en lugar de centrarse en detalles técnicos.

- **Priorización**: Permiten priorizar el trabajo basado en el valor que cada historia aporta al usuario o al negocio.
- **Fragmentación del trabajo**: Ayudan a dividir grandes funcionalidades en piezas manejables y entendibles.



4.3 Escribiendo Historias de Usuario

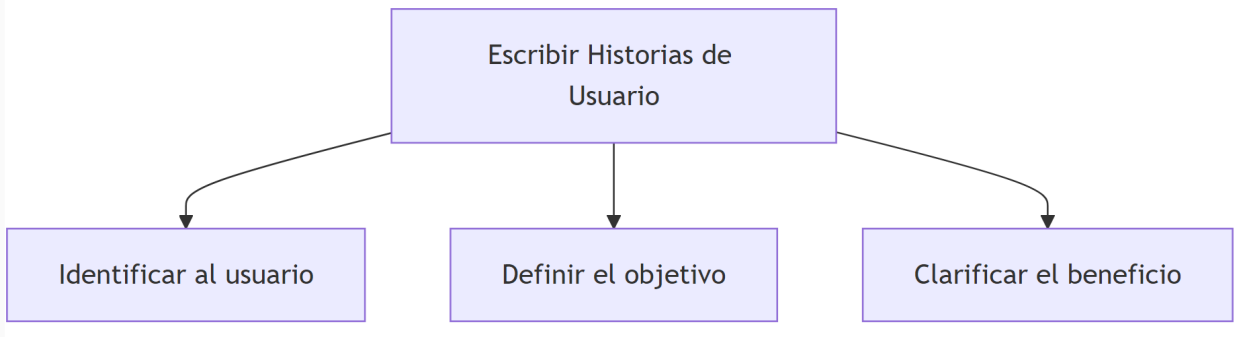
Para escribir una buena **historia de usuario**, es importante seguir el formato estándar: **"Como [rol], quiero [objetivo] para [beneficio]"**. Esto asegura que la historia esté centrada en el usuario y sus necesidades.

Al escribir historias de usuario:

1. **Identificar al usuario**: Determinar quién utilizará la funcionalidad (por ejemplo, un cliente, administrador, etc.).
2. **Definir el objetivo**: Describir lo que el usuario necesita o quiere hacer.
3. **Clarificar el beneficio**: Explicar el valor o resultado que obtiene el usuario al completar la acción.

Ejemplo:

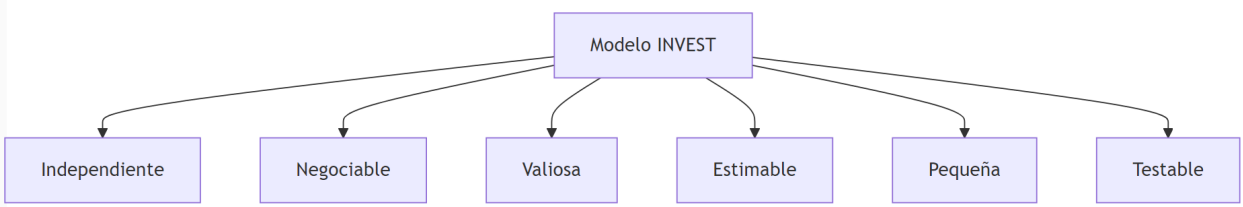
- **Como** administrador, **quiero** generar informes de ventas mensuales, **para** analizar el rendimiento del negocio.



4.4 Modelo INVEST

El modelo **INVEST** es una guía que asegura que las historias de usuario sean efectivas. Cada historia debe cumplir con los siguientes criterios:

- **I**: **Independiente**. La historia debe poder desarrollarse de manera aislada.
- **N**: **Negociable**. Debe poder discutirse y modificarse.
- **V**: **Valiosa**. Debe aportar valor al usuario o al negocio.
- **E**: **Estimable**. Debe ser posible estimar el esfuerzo necesario.
- **S**: **Small** (Pequeña). La historia debe ser lo suficientemente pequeña para completarse en un sprint.
- **T**: **Testable**. Debe haber una manera clara de verificar si se ha completado con éxito.



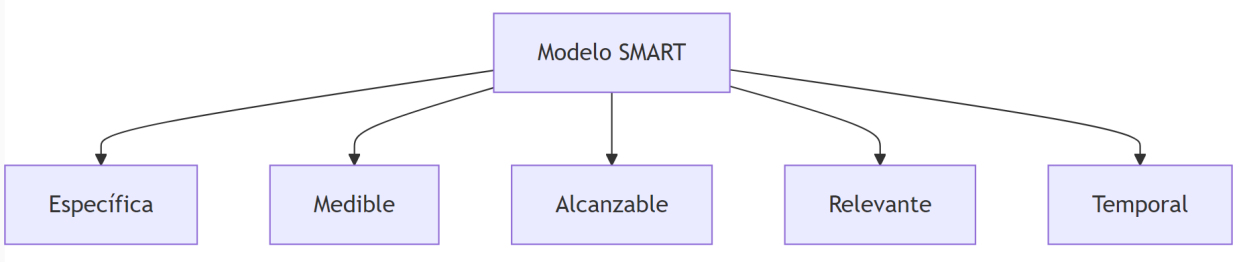
Ejemplo de historia de usuario **INVEST**:

- **Como** usuario premium, **quiero** acceder a contenido exclusivo, **para** disfrutar de beneficios adicionales. Esta historia es pequeña, independiente de otras funcionalidades, aporta valor y puede ser estimada.

4.5 Modelo SMART

El modelo **SMART** es otra técnica para crear historias de usuario efectivas. Cada historia debe ser:

- **S**: **Específica** (Specific). Debe describir claramente lo que se necesita.
- **M**: **Medible** (Measurable). Debe tener un criterio de éxito claro.
- **A**: **Alcanzable** (Achievable). Debe ser realista y posible de completar en un sprint.
- **R**: **Relevante** (Relevant). Debe ser importante para los objetivos del producto.
- **T**: **Temporal** (Time-bound). Debe poder completarse en un tiempo determinado, idealmente dentro de un sprint.



Ejemplo de historia de usuario **SMART**:

- **Como** usuario, **quiero** poder restablecer mi contraseña en menos de 5 minutos, **para** acceder rápidamente a mi cuenta en caso de olvidar la contraseña.

Tanto los modelos **INVEST** como **SMART** ayudan a que las historias de usuario sean claras, accionables y centradas en el valor que aportan al usuario final o al negocio.

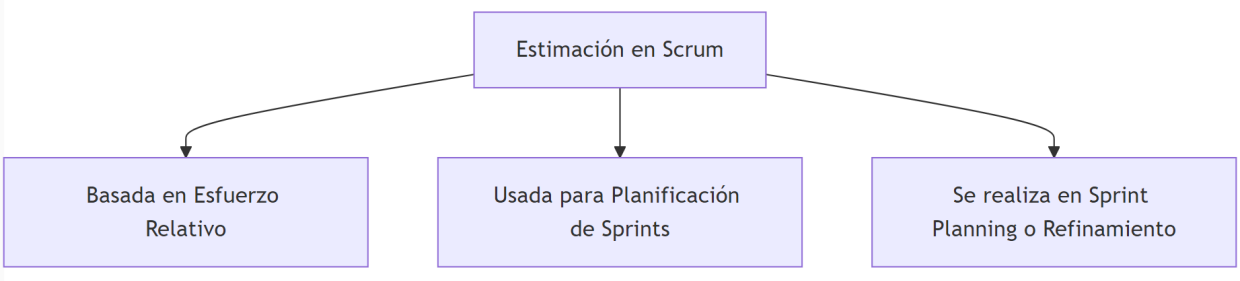
5. Taller de Estimación

5.1 Estimación en Scrum

En **Scrum**, la **estimación** es una práctica clave para planificar y priorizar el trabajo dentro del equipo. En lugar de utilizar estimaciones tradicionales basadas en el tiempo, como las horas hombre (HH), Scrum utiliza métodos basados en el esfuerzo relativo y la complejidad,

permitiendo a los equipos estimar la dificultad de una tarea o historia de usuario sin atarse a una cantidad fija de tiempo.

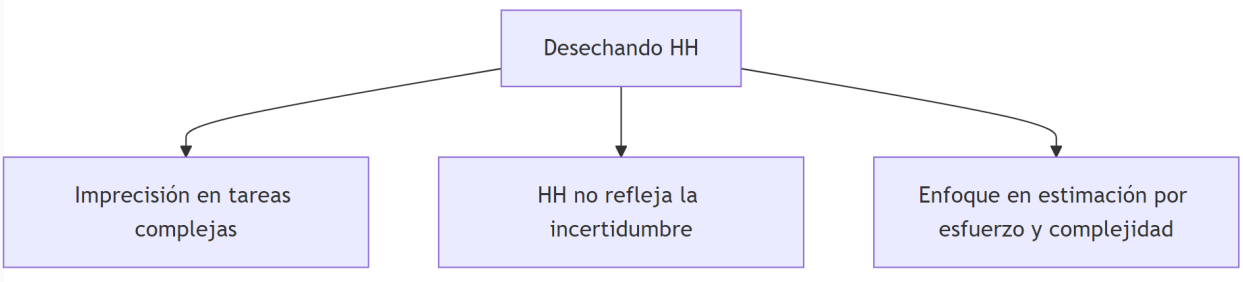
Las estimaciones se realizan durante la **Sprint Planning** o en sesiones de refinamiento del **Product Backlog**, donde el equipo discute y asigna un valor de estimación a cada historia de usuario o tarea.



5.2 Desechando la estimación en Horas Hombre (HH)

En Scrum, las **horas hombre (HH)** no son el enfoque preferido para la estimación, ya que pueden generar una falsa expectativa de precisión. Las tareas suelen implicar incertidumbre y cambios, y las estimaciones en tiempo fijo no reflejan bien la complejidad o los riesgos asociados.

En lugar de utilizar HH, Scrum utiliza técnicas como **puntos de historia** o **tallas** para medir el esfuerzo y la complejidad. Estas técnicas permiten que las estimaciones sean más flexibles y basadas en la capacidad del equipo, eliminando la presión de cumplir con un tiempo exacto.



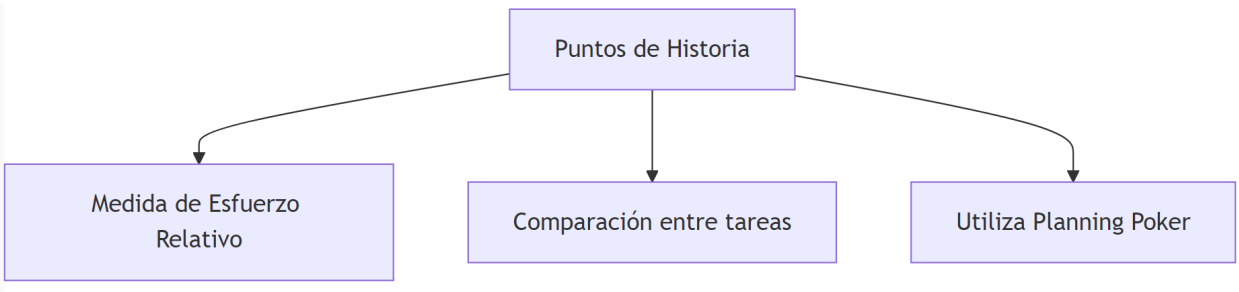
5.3 Modelos de Estimación Ágiles

Existen varios modelos de estimación utilizados en Scrum y otras metodologías ágiles que ayudan a los equipos a evaluar el esfuerzo de las tareas de manera colaborativa y precisa.

5.3.1 Puntos de Historia

Los **puntos de historia** son una unidad de medida que los equipos utilizan para estimar la complejidad de una historia de usuario o tarea. No están relacionados con el tiempo, sino con el esfuerzo relativo requerido para completar el trabajo. Los equipos comparan las tareas y asignan puntos de historia basados en su dificultad en relación con otras tareas conocidas.

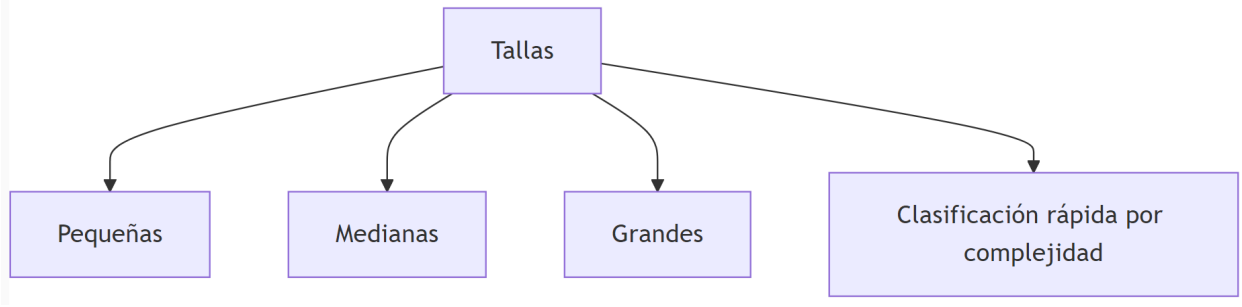
Planning Poker es una técnica común para estimar puntos de historia, donde cada miembro del equipo propone una estimación y se llega a un consenso después de discutir las diferencias.



5.3.2 Tallas

El método de **tallas** es otra forma de estimación ágil, donde las tareas se clasifican en categorías de "tamaño" como pequeñas, medianas o grandes. Cada tamaño representa un rango de esfuerzo o complejidad, lo que permite que el equipo estime de manera rápida sin discutir detalles precisos.

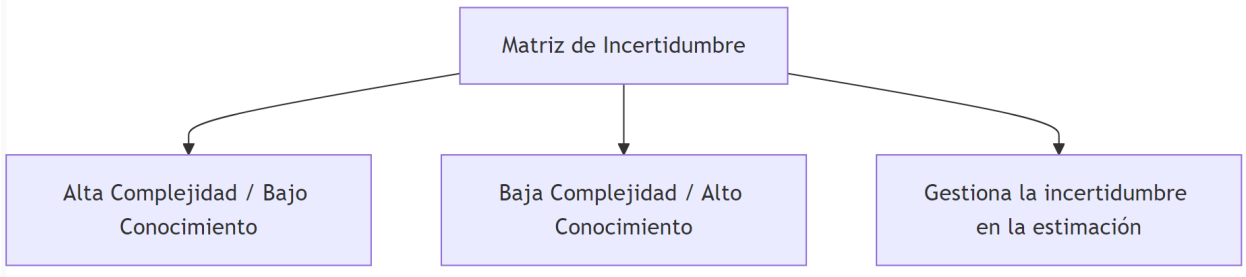
Este método es útil cuando el equipo necesita hacer estimaciones rápidas, y las tareas pueden agruparse fácilmente según su complejidad general.



5.3.3 Matriz de Incertidumbre

La **matriz de incertidumbre** es un enfoque que ayuda a los equipos a gestionar tareas que tienen un alto nivel de incertidumbre en cuanto a su implementación. Las tareas se clasifican según dos dimensiones: **complejidad** y **conocimiento**. Esto permite al equipo identificar tareas que requieren más investigación o experimentación antes de ser completamente estimadas.

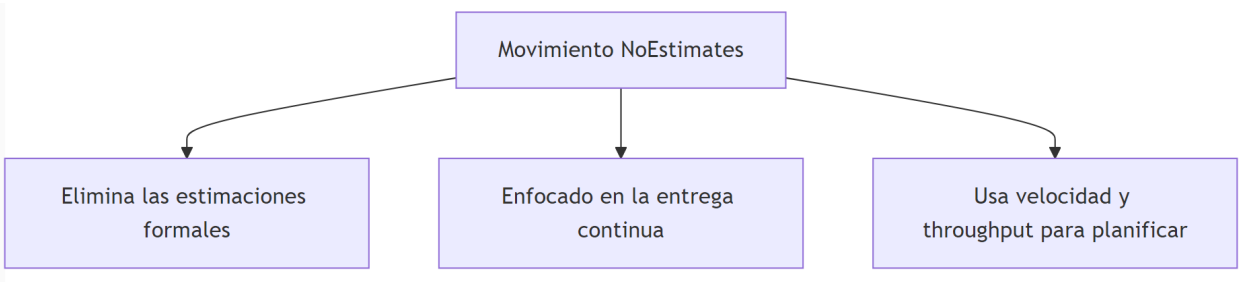
- **Alta complejidad y bajo conocimiento**: Estas tareas necesitan ser exploradas antes de ser estimadas.
- **Baja complejidad y alto conocimiento**: Estas tareas pueden ser estimadas rápidamente.



5.4 Movimiento NoEstimates

El **movimiento NoEstimates** propone que los equipos pueden ser más efectivos si eliminan las estimaciones por completo. Este enfoque sugiere que, en lugar de gastar tiempo en estimar el esfuerzo de cada tarea, el equipo debe enfocarse en entregar trabajo de manera continua y utilizar métricas como la **velocidad** o el **throughput** para medir su capacidad y planificar futuros sprints.

La idea detrás de NoEstimates es que el esfuerzo dedicado a las estimaciones a menudo no produce resultados precisos, y que los equipos pueden planificar mejor basándose en el historial real de entrega.



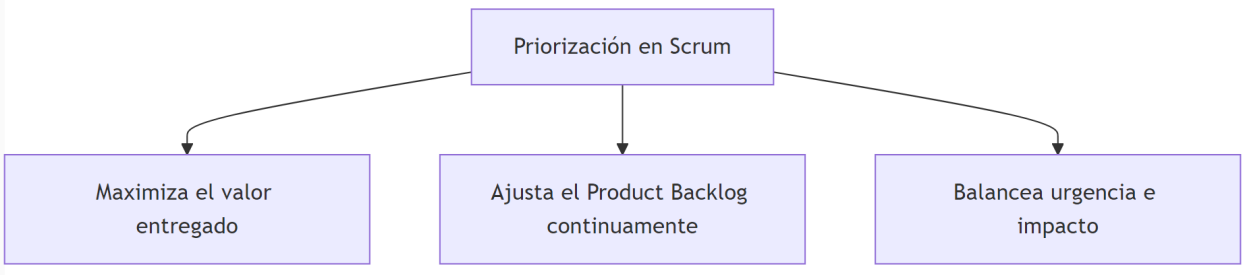
Este enfoque aún genera controversia, ya que algunos equipos prefieren mantener una estructura formal de estimación, mientras que otros consideran que eliminar estimaciones fomenta una mayor agilidad.

6. Priorización

6.1 En qué consiste la práctica de priorización

La **priorización** en **Scrum** es el proceso mediante el cual el **Product Owner** y el equipo deciden qué elementos del **Product Backlog** deben ser abordados primero, basándose en su valor, urgencia, impacto y complejidad. El objetivo principal de la priorización es maximizar el valor entregado al cliente en cada sprint, asegurando que los recursos del equipo se utilicen de la manera más eficiente posible.

La priorización es una actividad continua, ya que el backlog debe ajustarse conforme cambian las necesidades del negocio, se obtienen nuevas ideas o se recibe retroalimentación del cliente.

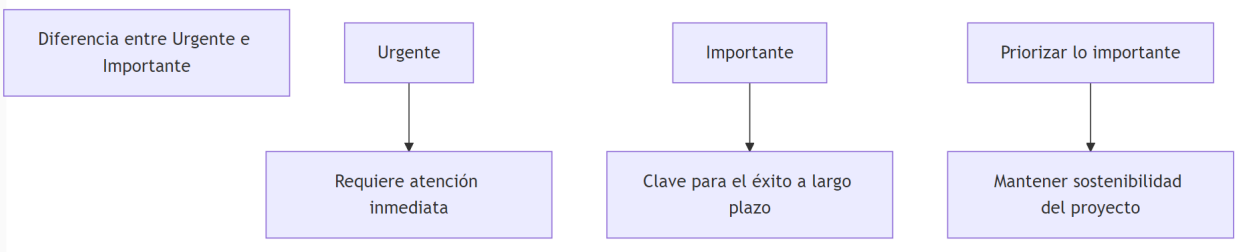


6.2 Diferenciar lo Urgente de lo Importante

Una habilidad crucial en la priorización es saber diferenciar entre lo **urgente** y lo **importante**.

- **Lo urgente**: Se refiere a las tareas que requieren atención inmediata, pero no necesariamente aportan un valor significativo a largo plazo.
- **Lo importante**: Son tareas que, aunque no requieran atención inmediata, son clave para el éxito a largo plazo del proyecto.

La clave está en priorizar lo importante sobre lo urgente cuando sea posible, ya que trabajar solo en lo urgente puede llevar a decisiones apresuradas y a descuidar el crecimiento y la sostenibilidad del producto.

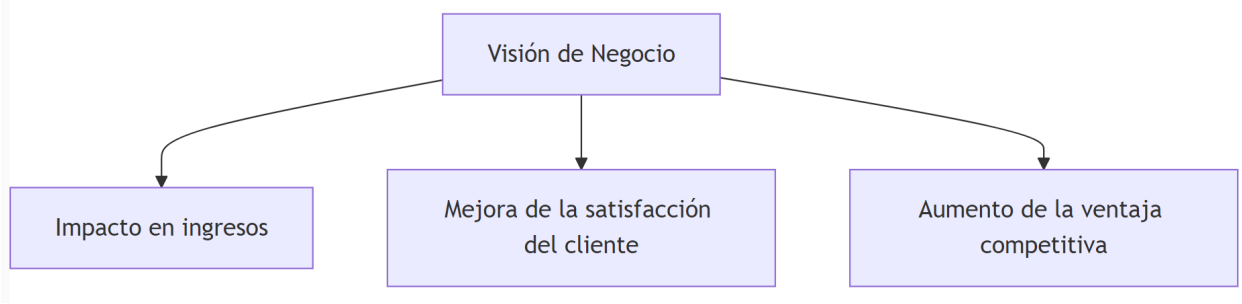


6.3 Modelos de Priorización

Existen varios modelos y técnicas que ayudan a los equipos a priorizar de manera efectiva. Estos modelos se basan en diferentes criterios como el valor de negocio, la complejidad o el impacto.

6.3.1 Visión de Negocio

Este enfoque de priorización pone el foco en el **valor de negocio** que cada funcionalidad aporta. El **Product Owner** evalúa cada tarea en términos de cómo contribuye a los objetivos estratégicos del negocio. Las tareas con mayor impacto en los ingresos, la satisfacción del cliente o la ventaja competitiva reciben mayor prioridad.

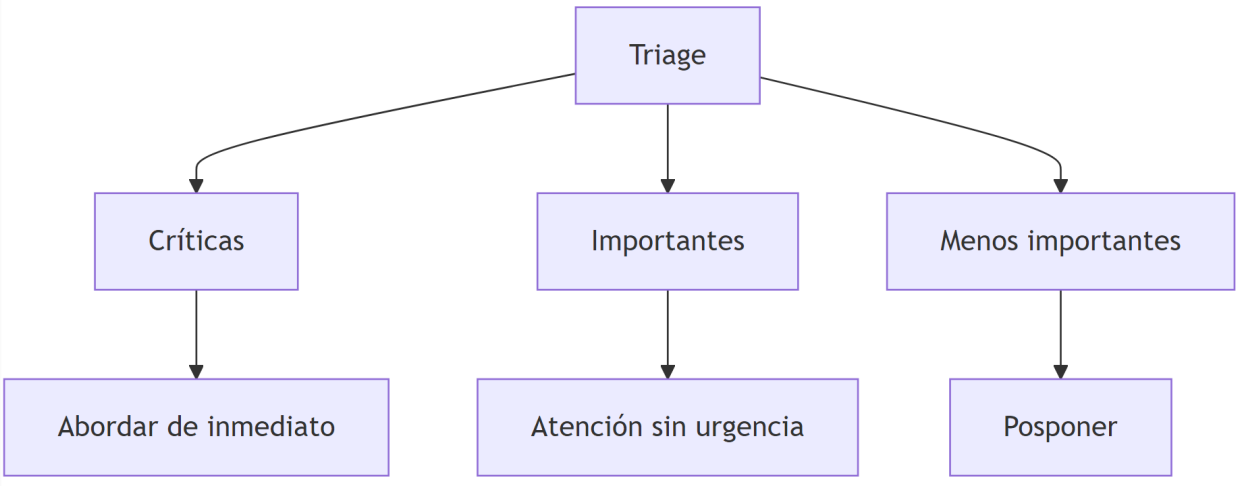


6.3.2 Triage

El **triage** es una técnica derivada de la práctica médica, donde las tareas se agrupan en categorías según su criticidad. Las categorías son:

1. **Críticas**: Deben ser abordadas de inmediato, ya que impactan directamente en la funcionalidad básica del producto.
2. **Importantes**: Necesitan atención, pero no son urgentes.
3. **Menos importantes**: Pueden ser pospuestas sin afectar gravemente el desarrollo del producto.

Esta técnica es útil cuando el equipo tiene recursos limitados y necesita enfocarse en lo más crítico primero.

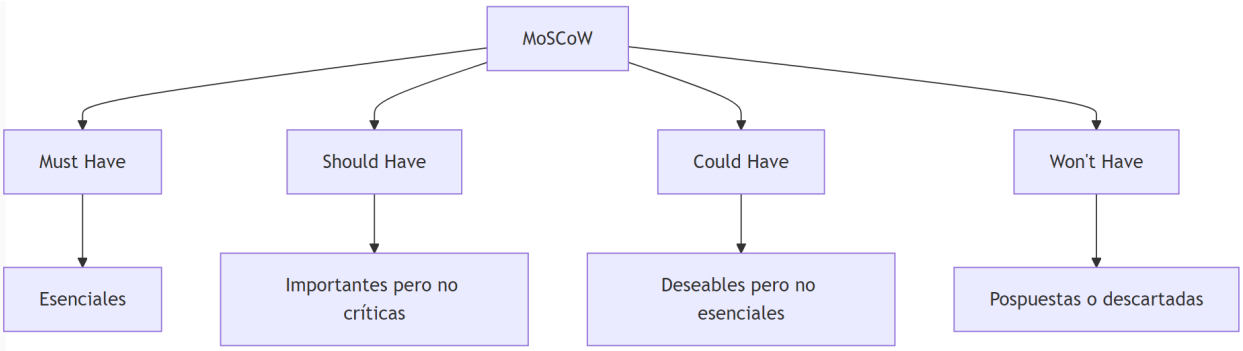


6.3.3 Moscow

El método **MoSCoW** es una técnica popular de priorización que divide las tareas en cuatro categorías:

1. **Must Have (Debe Tener)**: Funcionalidades que son esenciales para el éxito del producto.
2. **Should Have (Debería Tener)**: Funcionalidades importantes pero no críticas.
3. **Could Have (Podría Tener)**: Funcionalidades deseables pero no esenciales.
4. **Won't Have (No Tendrá)**: Funcionalidades que no se incluirán en el sprint actual, pero podrían considerarse en el futuro.

El **Product Owner** utiliza esta técnica para tomar decisiones rápidas sobre qué elementos abordar en cada sprint.

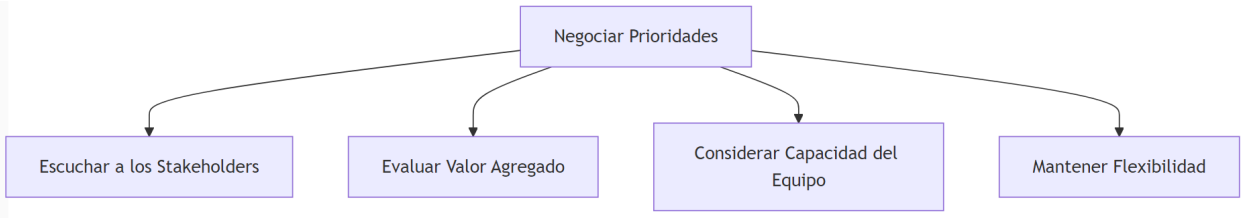


6.4 Cómo Negociar Prioridades

La **negociación de prioridades** es una habilidad importante en Scrum, especialmente entre el **Product Owner** y los stakeholders. A medida que se identifican nuevas prioridades, el Product Owner debe equilibrar los deseos de los stakeholders con las limitaciones del equipo y el objetivo de maximizar el valor.

Al negociar prioridades, es esencial:

- **Escuchar a los stakeholders**: Entender sus necesidades y preocupaciones.
- **Evaluar el valor agregado**: Priorizar las funcionalidades que aportan más valor al negocio o al cliente.
- **Considerar la capacidad del equipo**: Asegurarse de que las tareas sean realistas y alcanzables en el sprint actual.
- **Mantener la flexibilidad**: Adaptar las prioridades conforme evolucionen las necesidades del proyecto.



Esta negociación constante es esencial para que el equipo trabaje en las tareas más valiosas y urgentes, mientras se mantienen alineados con la visión del producto y las expectativas del negocio.

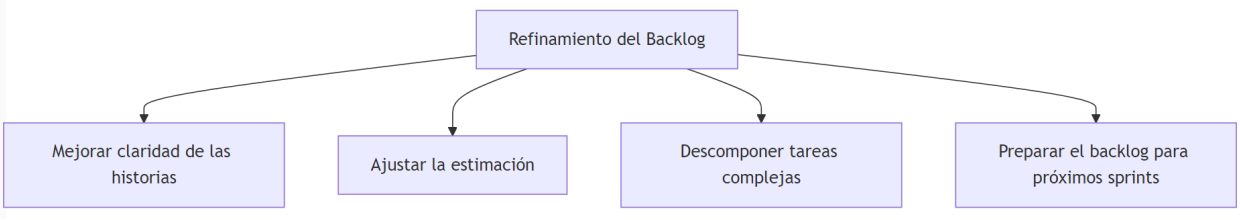
7. Refinamiento del Backlog

7.1 Para qué sirve el Refinamiento del Backlog

El **refinamiento del backlog** es una práctica en **Scrum** que se utiliza para mejorar la claridad, comprensión y priorización de los elementos en el **Product Backlog**. Durante estas sesiones, el equipo revisa las historias de usuario y tareas pendientes, las descompone en ítems más manejables, ajusta su estimación y priorización, y resuelve dudas sobre los requerimientos.

Esto garantiza que los elementos más importantes estén listos para ser trabajados en los próximos sprints.

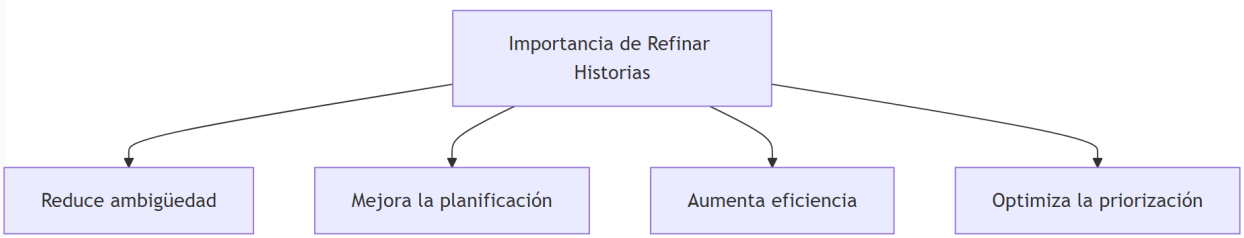
El refinamiento del backlog ayuda a evitar sorpresas durante la **Sprint Planning**, asegurando que el equipo pueda comenzar a trabajar sin contratiempos.



7.2 Importancia de Refinar Historias

Refinar las historias de usuario es crucial para asegurar que cada ítem del **Product Backlog** esté listo para ser implementado. Una historia de usuario bien refinada:

- **Reduce la ambigüedad**: Ayuda a que el equipo comprenda claramente lo que se necesita desarrollar.
- **Mejora la planificación**: Al tener historias bien definidas y descompuestas, el equipo puede hacer una mejor estimación y planificación.
- **Aumenta la eficiencia**: Al identificar posibles problemas o desafíos durante el refinamiento, el equipo puede evitarlos antes de que se conviertan en bloqueadores durante el sprint.
- **Optimiza la priorización**: Las historias refinadas permiten que el **Product Owner** priorice de manera más precisa, enfocándose en el valor que cada historia aporta.

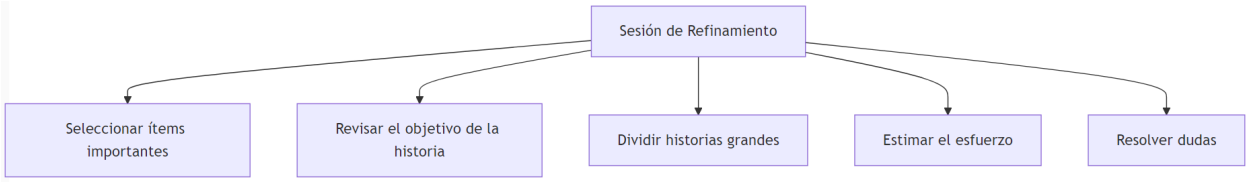


7.3 Cómo llevar sesiones de Refinamiento

Las sesiones de **refinamiento del backlog** son reuniones en las que el equipo de desarrollo, el **Product Owner**, y en ocasiones otros stakeholders, colaboran para revisar los elementos del backlog. Estas sesiones suelen realizarse de manera continua, no necesariamente dentro del sprint, y su objetivo es mantener siempre un conjunto de ítems listos para el trabajo.

Pasos para una sesión efectiva de refinamiento:

1. **Seleccionar ítems importantes**: El **Product Owner** selecciona los elementos más prioritarios para ser refinados.
2. **Revisar el objetivo de la historia**: El equipo y el Product Owner revisan y aseguran que la historia de usuario esté clara.
3. **Dividir historias grandes**: Si una historia es muy grande o compleja, el equipo la descompone en tareas más pequeñas que puedan completarse dentro de un sprint.
4. **Estimar el esfuerzo**: El equipo utiliza métodos como **puntos de historia** para estimar la complejidad y el esfuerzo necesario.
5. **Resolver dudas**: Se resuelven preguntas sobre los requisitos o los detalles técnicos de las historias.

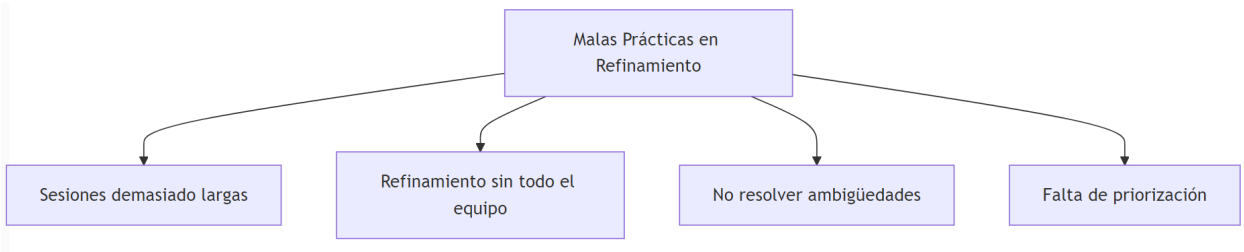


7.4 Malas prácticas en el Refinamiento

Existen ciertas malas prácticas que pueden afectar la efectividad del refinamiento del backlog. Estas incluyen:

- **Refinamientos demasiado largos**: Si las sesiones de refinamiento se extienden demasiado, pueden perder su eficacia. El refinamiento debe ser continuo y no consumir más del 10% del tiempo del sprint.
- **Refinamiento sin todo el equipo**: El refinamiento debe involucrar tanto al equipo de desarrollo como al **Product Owner**. Si falta alguno de estos actores, el refinamiento puede ser inefectivo.

- **No resolver ambigüedades**: El propósito del refinamiento es aclarar las historias. Si las historias permanecen vagas después de la sesión, se pierde el objetivo.
- **Falta de priorización adecuada**: Refinar historias que no son prioritarias puede ser una pérdida de tiempo. El equipo debe enfocarse en las historias más importantes que se trabajarán en los próximos sprints.



Evitar estas malas prácticas asegura que las sesiones de refinamiento sean productivas y permitan que el equipo avance de manera eficiente, preparando el **Product Backlog** para los próximos sprints.

Material de Referencia

Libros:

- **"Scrum: The Art of Doing Twice the Work in Half the Time"** de Jeff Sutherland
  Escrito por uno de los creadores de Scrum, este libro explora en profundidad los principios y prácticas de Scrum, cómo implementarlos en equipos de trabajo y los beneficios de la agilidad. Ideal para comprender los fundamentos de Scrum y su impacto en la productividad.

- **"Agile Estimating and Planning"** de Mike Cohn
  Este libro ofrece una visión clara sobre la estimación y planificación en entornos ágiles. Mike Cohn proporciona herramientas prácticas para mejorar la precisión de las estimaciones y cómo gestionar proyectos complejos en Agile. Es muy útil para entender temas como los puntos de historia y la priorización.

- **"Essential Scrum: A Practical Guide to the Most Popular Agile Process"** de Kenneth S. Rubin

Esta guía completa sobre Scrum cubre todos los aspectos del marco, desde roles y ceremonias hasta artefactos y métricas. Es ideal tanto para principiantes como para equipos experimentados que buscan optimizar su implementación de Scrum.

- **"User Story Mapping: Discover the Whole Story, Build the Right Product"** de Jeff Patton
  Este libro explica cómo utilizar historias de usuario de manera efectiva para mejorar la comprensión del producto y alinear a los equipos en torno a un objetivo común. Es una referencia fundamental para la creación de **historias de usuario** y la planificación del **MVP** en proyectos ágiles.

Enlaces a Recursos Online:

- **Scrum Guide - Scrum.org**: La guía oficial de Scrum, escrita por los creadores Jeff Sutherland y Ken Schwaber. Es el documento base para entender el marco Scrum, sus roles, eventos y artefactos. Es un recurso esencial para cualquier equipo que quiera implementar Scrum correctamente.

- **Manifesto for Agile Software Development (agilemanifesto.org)**: El sitio oficial del **Manifiesto Ágil**, donde se detallan los valores y principios que guían a todas las metodologías ágiles, incluyendo Scrum. Es un recurso clave para entender la filosofía detrás de Agile.

- **Mountain Goat Software - Mike Cohn**: Mike Cohn, uno de los principales expertos en Scrum, ofrece una amplia gama de artículos y guías prácticas sobre **estimación ágil**, planificación de sprints, y creación de historias de usuario.

- **Atlassian Agile Coach**: Atlassian proporciona una excelente serie de guías sobre temas ágiles, incluyendo **Scrum**, **Kanban**, **historias de usuario**, **priorización** y **refinamiento del backlog**. Es una referencia útil para aprender y mejorar las prácticas ágiles en equipos de desarrollo.

Videos Recomendados:

- **Scrum in Under 10 Minutes - Scrum.org**: Un video corto y conciso que explica los conceptos fundamentales de Scrum en menos de 10 minutos. Ideal para principiantes que necesitan una visión rápida y clara de cómo funciona Scrum.

- [Agile Product Ownership in a Nutshell - Henrik Kniberg]: Un video que detalla el rol del **Product Owner** en Scrum, explicando cómo manejar el backlog, priorizar historias y entregar valor de forma continua. Es una excelente introducción al trabajo del Product Owner en proyectos ágiles.

CORFO
BecasCapital Humano

Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 7

FUNDAMENTOS DE DESARROLLO AGILE