The Scrum Map

Desde CertMind, te damos la bienvenida a este libro, tu aliado en el fascinante mundo del desarrollo de productos bajo una filosofía ágil. Felicitaciones por tomar este paso significativo en beneficio de tus equipos y tu organización. Este libro no es solo para leer; es para interactuar, aplicar y reflexionar. Ya sea que lo estés consultando en línea o en PDF, utilízalo como una herramienta dinámica en tu camino hacia el dominio de Scrum.Aquí encontrarás un complemento esencial a 'The 2020 Scrum Guide' (https://scrumguides.org), enriquecido con explicaciones prácticas, ejemplos y ejercicios que elevarán tu comprensión de Scrum. Basamos nuestra guía en años de experiencia, colaboraciones con expertos, estudiantes y voluntarios. Nos inspiramos tanto en la guía oficial de Scrum como en el destacado 'The New New Product Development Game' de la Harvard Business Review. Así que, adelante, únete a nosotros en este viaje apasionante de transformar ideas en productos excepcionales.

1. Introducción: De la filosofía ágil a la práctica con Scrum

Este capítulo inicial te introduce en el mundo de la agilidad como filosofía de trabajo, sentando las bases para entender cómo Scrum se inserta en este contexto. Exploraremos los fundamentos de ambos conceptos para darte una comprensión sólida antes de abordar aspectos más específicos. Puntos clave de este capítulo: - Descubre la esencia de la agilidad y por qué importa. - Aprende cómo la agilidad impulsa la creación de productos exitosos. - Entiende el lugar que ocupa Scrum en el mundo ágil. - Hazte amigo de los términos de Scrum que serán tus compañeros de viaje en este libro.

1.1 - ¿Qué es la agilidad y por qué se habla de ella en Scrum?

En el mundo actual, caracterizado por su dinamismo y constantes cambios, la agilidad se ha convertido en una filosofía esencial que promueve la adaptabilidad y la entrega continua de valor.

Antes de adentrarnos en el marco de trabajo Scrum, una de las herramientas más prominentes para implementar las prácticas ágiles, es fundamental que tengamos una sólida comprensión de lo que encarna la agilidad.

En este capítulo, exploraremos qué significa ser ágil, los beneficios que aporta, y los desafíos que podrías enfrentar al adoptar este enfoque. Esta base nos preparará perfectamente para luego entender cómo la cultura ágil y Scrum pueden colaborar en la transformación no solo de proyectos, sino también de organizaciones completas.

> **Caso de estudio: Adopción de la agilidad para mejorar las empresas**
> Antes de adoptar prácticas ágiles, "People Information" se encontraba en una situación donde los proyectos demoraban mucho en completarse y a menudo excedían los presupuestos establecidos. La decisión de abrazar la agilidad y específicamente Scrum, fue un cambio estratégico crucial; Al enfocarse en ser más ágiles, empezaron a trabajar de una manera más colaborativa e innovadora, lo que les permitió responder rápidamente a los cambios y mantener una alta calidad en su entrega de productos. Este cambio no solo optimizó sus procesos internos sino que también mejoró la satisfacción de sus clientes al entregar soluciones que verdaderamente atendían sus necesidades actuales.

Scrum y su relación con la agilidad

Scrum es uno de los marcos de trabajo más reconocidos para adoptar la agilidad en equipos y organizaciones. Es un marco de trabajo que se enfoca principalmente en el desarrollo de productos, permitiendo que los equipos trabajen de manera iterativa e incremental. Esto significa

que, en lugar de planificar todo el trabajo desde el principio, el trabajo se divide en pequeñas partes manejables, entregando valor continuo y permitiendo ajustes basados en el feedback real y las condiciones cambiantes del mercado.

> Adoptar Scrum puede ser una estrategia eficaz para ayudar a las organizaciones a navegar por la complejidad del mundo actual, permitiendo entregas más rápidas y soluciones de mayor calidad que realmente resuenen con las necesidades de los clientes.

Scrum adopta los principios y valores contenidos en el Manifiesto Ágil (https://agilemanifesto.org/), lo que facilita la creación de productos de alta calidad, fomentando la colaboración y la respuesta rápida a los cambios. Al integrar Scrum en su enfoque hacia la agilidad, las organizaciones pueden esperar una mayor satisfacción del cliente, equipos más colaborativos y soluciones que verdaderamente atienden las necesidades actuales y emergentes del mercado.

□□ Tip para el lector

Promueve la formación continua en métodos ágiles dentro de tu organización. Considera establecer un programa de formación interno o aprovecha los cursos disponibles en plataformas reconocidas para desarrollar las habilidades necesarias en tu equipo. La formación constante facilita una transición más suave hacia las prácticas ágiles y ayuda a mitigar la resistencia al cambio

y entonces ... ¿Qué es la agilidad?

La agilidad es una mejor manera de resolver problemas complejos, con enfoque en la generación de espacios de colaboración, confianza e innovación.

La habilidad para aplicar estas técnicas de trabajo está determinada por el hecho de que los equipos de trabajo sean multifuncionales, enfocados en el cliente y autogestionados, lo cual contribuye de forma eficiente a generar estrategias enfocadas al cambio, permitiendo identificar rápidamente la mejora en los procesos; haciendo que se defina y construya el producto de acuerdo con las necesidades de la empresa y del cliente. Por lo general todos los métodos ágiles están diseñados para que el equipo de trabajo realice el trabajo de manera incremental e iterativamente.

> La adopción de prácticas ágiles puede ser un verdadero catalizador para fomentar la innovación y mejorar la entrega de productos en una organización.

Beneficios de practicar la agilidad

- **Mayor satisfacción del cliente:** El cliente está continuamente involucrado en el proceso de desarrollo y puede ver y guiar la evolución del producto.
- **Entrega más rápida:** Al desglosar el desarrollo del producto en iteraciones manejables, se puede entregar valor más rápidamente y con mayor frecuencia.
- **Mayor calidad del producto:** Con la retroalimentación continua y la posibilidad de realizar cambios durante el proceso, es más probable que el producto final cumpla con las expectativas del cliente.
- **Aumento de la moral del equipo:** Los equipos ágiles suelen estar más motivados y comprometidos con el producto, ya que tienen más control sobre su trabajo.
- **Mejor manejo de la incertidumbre y el cambio:** Las prácticas ágiles se adaptan bien a los cambios, ya sea en los requisitos del cliente o en las condiciones del mercado.

> **Caso de Estudio: Renovación Ágil en TechSolutions** "TechSolutions", una firma de tecnología, enfrentaba entregas tardías y descontento de clientes. Para remediar esto, optaron por adoptar Scrum en uno de sus equipos centrales. Esta transición rápidamente promovió una colaboración intensificada y elevó la moral del equipo, facilitando entregas más rápidas y una mejor calidad de producto; La integración continua del cliente en el proceso aseguró que los productos finales estuvieran más alineados con sus necesidades cambiantes, elevando así la satisfacción del cliente. Este cambio, aunque inicialmente implementado en un solo equipo, inspiró una transformación ágil a lo largo de la empresa, consolidando a "TechSolutions" como un testimonio de éxito y eficiencia en la industria.

Desafíos al adoptar prácticas ágiles

> Ser ágil va más allá de simplemente adoptar nuevas herramientas o técnicas; es un cambio cultural que pone un fuerte énfasis en la colaboración, la comunicación y la adaptabilidad.

- **Cambio cultural/resistencia al cambio:** La adopción de prácticas ágiles puede requerir un cambio cultural significativo, especialmente en organizaciones más tradicionales.
- **Requiere compromiso y colaboración continua del cliente:** No todas las organizaciones o clientes están preparados para este nivel de involucramiento.
- **Es difícil de planificar a largo plazo:** Debido a su naturaleza adaptable, la planificación a largo plazo (años) puede ser un desafío usando marcos ágiles.
- **Falta de experiencia o formación en métodos ágiles:** Si los miembros del equipo no están familiarizados con los marcos ágiles, pueden tener dificultades para adaptarse a las nuevas formas de trabajar.

1.2 - La cultura ágil: Cuando mejorar procesos no es suficiente

En un mundo donde la competencia se intensifica día a día, adoptar una cultura ágil ya no es una preferencia, sino una necesidad para las empresas que aspiran a destacarse. La Cultura Ágil, más que un conjunto de prácticas, representa una revolución en la forma en que los equipos piensan y operan, poniendo en primer lugar la creación de valor y la satisfacción del cliente. Acompáñanos mientras desentrañamos los conceptos clave de "Ser Ágil" y "Hacer Ágil", elementos cruciales para abrazar la agilidad en tu organización.

> Una cultura ágil es donde la innovación se encuentra con la eficiencia, creando un espacio donde las ideas florecen y se realizan sin obstáculos.

La esencia de la Cultura Ágil

La cultura ágil, también conocida como Cultura Agile (incluso muy similar a lo que algunos llaman "Agile Mindset"), implica una transformación radical en la forma de trabajar. Se trata de adoptar estrategias que reduzcan la complejidad y centren los esfuerzos en crear productos o servicios que respondan directamente a las necesidades del cliente.

El propósito es maximizar el potencial de los equipos de trabajo, permitiéndoles alcanzar resultados destacados en plazos más breves y dinámicos, con un enfoque sólido en la excelencia, la competitividad, y la colaboración integral.

Ser Ágil vs. Hacer Ágil: Un equilibrio necesario

Hacer Ágil

"Hacer Ágil" es el acto de aplicar técnicas y estrategias específicas del ámbito ágil. En esta fase, nos centramos en incorporar eventos estructurados y definidos que faciliten el desarrollo fluido de un producto. Aquí, se integran herramientas específicas y se adoptan responsabilidades definidas para gestionar el trabajo de una manera más eficiente y productiva.

Ser Ágil

En contraste, "Ser Ágil" va más allá de las prácticas tangibles, sumergiéndose en la transformación de la mentalidad de cada integrante del equipo. Se trata de cultivar un enfoque que valora la colaboración, la adaptabilidad y un compromiso constante con el cliente. Esta faceta busca inculcar valores y principios ágiles en cada persona, fomentando una cultura de mejora continua y una predisposición a la innovación.

> En nuestra ilustración, el cerebro representa de manera simbólica la dualidad de la cultura ágil. En el **hemisferio izquierdo, se destaca el "Hacer Ágil"**, que engloba la estructura y las prácticas concretas que guían el trabajo diario, sirviendo como el pilar racional y metódico de la gestión ágil. Por su parte, el **hemisferio derecho resalta el "Ser Ágil"**, un espacio donde reside la mentalidad y los valores que deben permear en cada individuo y equipo, fomentando una actitud de colaboración, innovación y enfoque centrado en el cliente. Esta representación **simboliza la armonía y el equilibrio necesarios entre la adopción de prácticas ágiles estructuradas y el fomento de una mentalidad ágil, demostrando que una verdadera cultura ágil prospera cuando ambos hemisferios trabajan en una sintonía perfecta**.

> Ten presente que debería existir tanto el ser, como el hacer en una correcta adopción de Scrum, así que no debes centrarte sólo en una o en otra.

> **Caso de estudio: cuando cambiar los procesos no es suficiente** People Information LLC, una firma especializada en el outsourcing de recursos humanos, decide mejorar sus procesos usando Scrum. En su fase inicial, se enfocaron en estructurar y organizar sesiones de trabajo y asignar responsabilidades claras (Hacer Ágil), logrando una notable mejora en la eficiencia de sus procesos, Sin embargo, pronto se dieron cuenta de que para alcanzar un estado de agilidad verdadero, necesitaban ir más allá. Empezaron a fomentar una mentalidad ágil (Ser Ágil) en su equipo, promoviendo la colaboración, la comunicación constante y una fuerte orientación hacia el cliente. Esto no solo transformó su forma de trabajar sino que fomentó una

> cultura donde la innovación y la mejora continua eran el día a día, permitiendo una sintonía perfecta entre las metodologías ágiles y una mentalidad de crecimiento y adaptabilidad.

1.3 - Scrum: ¿Revolución o moda pasajera?

¿Es Scrum una revolución en el desarrollo de productos o simplemente una tendencia pasajera? Después de haber explorado la profundidad de la cultura ágil, ahora nos enfocamos en uno de sus marcos más destacadas: Scrum. Esta guía te proporcionará una visión detallada y clara de lo que es Scrum, sus orígenes y cómo puede ser un agente de cambio significativo en tu organización.

Sumérgete con nosotros en este análisis objetivo para determinar el verdadero valor de adoptar Scrum.

Aclaraciones de la guía

> Consejo para el lector: Es imperativo destacar que el libro "The Scrum Map" NO está restringido solo al ámbito del desarrollo de software. De hecho, se erige como una guía valiosa para cualquier profesión que aspire a implementar prácticas ágiles, facilitando así la gestión eficiente y la entrega de valor en una diversidad de campos profesionales. A través del "Scrum Map", los profesionales pueden trazar una ruta clara, estructurada y adaptable, que potencie la colaboración y la innovación en sus respectivos equipos.

- Esta es una guía llena de conceptos, ideas y técnicas para entender y adoptar Scrum de forma sencilla, esto no significa que sea un plan de implementación exacto, por lo que debe tomarse como consejos y no como una plantilla.
- En el desarrollo de esta guía el término “Producto” se refiere a productos, servicios y cualquier otro entregable que pueda surgir como resultado de un ejercicio de proyecto o desarrollo organizacional.
- Cuando los conceptos “Desarrollo” y “Desarrollar” se utilizan en esta guía de Scrum, se refieren al desarrollo de las actividades, no se ven limitados al campo del desarrollo de software.
- Este libro es la base fundamental para la obtención de las certificaciones CM-SFC (Scrum Fundamentals Certified), CM-SMC (Scrum Master Certified), CM-SDC (Scrum Developer Certified), CM-SPOC (Scrum Product Owner Certified) y CM-Scrum Expert que hacen parte del esquema de certificación en Scrum de CertMind.

Historia De Scrum

La historia de Scrum se puede rastrear desde 1986 en un artículo de la Harvard Business Review, “The New New Product Development Game” escrito por Hirotaka Takeuchi y Ikujiro Nonaka, en el que analizaron el enfoque que utilizaban compañías como Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox, y Hewlett-Packard para el desarrollo de sus productos (esto debido a que estas compañias destacaban por su capacidad de innovación y buen desarrollo de productos).

Durante sus investigaciones, se dieron cuenta que dichas empresas compartían seis características:

1. Inestabilidad incorporada
2. Equipos de proyectos autoorganizados
3. Fases de desarrollo superpuestas
4. Multi-aprendizaje
5. Control sutil
6. Transferencia organizativa de aprendizaje.

Este enfoque, visualizado como un proceso ágil y flexible, sirvió para rejuvenecer organizaciones estancadas, infundiendo creatividad y respuestas impulsadas por el mercado.

> El artículo original se puede leer en: https://hbr.org/1986/01/the-new-new-product-development-game

El camino de Scrum continuó en 1995 con Ken Schwaber y Jeff Sutherland, quienes consolidaron las ideas previas y presentaron una definición formal de Scrum en la conferencia OOPSLA de ese año, haciendo eco de su aprendizaje en empresas como Individual, Inc. y Fidelity Investments. Desde entonces, Scrum ha evolucionado bajo la custodia de Sutherland y Schwaber, sirviendo como un marco robusto y adaptable que complementa y potencia las estrategias de desarrollo de productos.

> La guía publicada por Ken Schwaber y Jeff Sitherland se puede consultar en: https://scrumguides.org/

... Pero y entonces ¿Qué es Scrum?

Scrum es un marco de trabajo (framework en inglés) que proporciona una orientación para organizar las personas y guiar su forma de trabajo para **construir productos de forma eficiente y creativa** con un objetivo principal: **entregar el máximo valor posible en el menor tiempo**,

esto se logra haciendo entregas parciales del producto, que también se conocen como incrementos de producto.

Cuando hablamos de Scrum, es importante destacar que pertenece a los **marcos de trabajo ágiles**, y que esto implica una transformación cultural, descubrimientos, experimentación, que al final se traducen en un cambio en la forma de desarrollar productos.

> Scrum no es simplemente un proceso, no es una técnica, o un método definitivo para hacer las cosas, es más bien un conjunto de muchas herramientas y guías para que puedas llevar la agilidad y el cambio que esto supone para la organización.

Usos de Scrum

Inicialmente concebido para la gestión y desarrollo de productos, el alcance de Scrum se ha ampliado considerablemente. Hoy en día, se ha evidenciado su utilidad en una variedad de contextos, tales como:

1. Investigar e identificar mercados viables, tecnologías, y capacidades. **Ejemplo:** Una empresa de energías renovables puede utilizar Scrum para investigar y desarrollar nuevas tecnologías de energía sostenible, ajustando su enfoque basándose en los hallazgos de cada incremento.
2. Desarrollo de mejoras a productos ya existentes. **Ejemplo:** Un equipo de desarrollo de software puede utilizar Scrum para implementar actualizaciones regulares basadas en el feedback de los usuarios.
3. Desarrollo de productos que requieren lanzamientos diariamente o tantas veces como sea posible. **Ejemplo:** Una plataforma de noticias online puede emplear Scrum para adaptar y actualizar su contenido varias veces al día, en respuesta a los acontecimientos mundiales en constante cambio.
4. Desarrollo y mantenimiento en ambientes cloud (en línea, con foco en seguridad, y servicios por demanda), facilitando una operatividad óptima para el uso de productos. **Ejemplo:** Un proveedor de servicios cloud puede emplear Scrum para mantener y actualizar su infraestructura de forma continua y ágil.
5. Mantenimiento y renovación de productos. **Ejemplo:** Una empresa manufacturera podría utilizar Scrum para la mejora continua de la línea de productos, permitiendo adaptaciones rápidas a las necesidades del mercado.
6. Scrum se ha demostrado especialmente efectivo en la transferencia de conocimiento iterativa e incrementalmente. Scrum es ampliamente utilizado para la construcción de productos y servicios. **Ejemplo:** Una organización educativa podría utilizar Scrum para desarrollar y mejorar cursos de formación, integrando feedback iterativo de los estudiantes para realizar mejoras continuas.

Conclusión

¿Es Scrum una revolución o una moda pasajera? A través de los años, Scrum ha demostrado ser una herramienta vital, propulsando la eficiencia y colaboración en diversos campos, más allá del desarrollo de software. Su flexibilidad y enfoque en la entrega de valor lo consolidan como una revolución en la gestión de productos y proyectos, no como una tendencia efímera.

A medida que continuamos explorando la profundidad de este marco de trabajo ágil, en nuestro próximo módulo, nos sumergiremos en el corazón de Scrum: el modelo core, donde descubrirás los componentes clave del marco:

- El equipo Scrum
- Los eventos
- Los artefactos

2. Modelo Core de Scrum

En este capítulo, hablaremos de las bases de Scrum, abriendo la caja para mostrar cada una de sus piezas fundamentales: el equipo, los artefactos, los eventos y la teoría que lo sostiene. Piensa en ello como los ingredientes básicos que necesitas para el desarrollo de productos exitosos. Puntos Clave de Este Capítulo: - Descubre quiénes componen el equipo en Scrum y cuál es su función. - Identifica los artefactos que actúan como herramientas de trazabilidad y planeación. - Reconoce los eventos que mantienen en marcha el proceso Scrum. - Comprende la teoría de Scrum que vincula todas las piezas juntas.

2.1 - El equipo Scrum

product-owner scrum-master developers

Image not found or type unknown Image not found or type unknown Image not found or type unknown

Un Equipo Scrum consiste en un Propietario del Producto (Product Owner), un Scrum Master y los Developers; y se trata de una unidad cohesionada de profesionales centrados en un objetivo a la vez: **el Objetivo de Producto (Product Goal).**

Los Equipos Scrum son auto-gestionados y multifuncionales. El modelo de Equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad.

Los Equipos Scrum entregan productos de forma iterativa e incremental, maximizando las oportunidades para poder obtener retroalimentación. Las entregas incrementales de producto “Terminado” aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto.

Tamaño del Equipo Scrum

El tamaño óptimo de un Equipo Scrum es **10 personas o menos,** esto le permite ser lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para poder completar una cantidad significativa de trabajo.

Consideraciones:

- No se recomienda tener menos de 3 miembros en el Equipo pues se reduce la interacción y resulta en baja productividad.
- Los Equipos más pequeños pueden encontrar limitaciones en cuanto a las habilidades necesarias durante un Sprint, haciendo que el Incremento pudiese no ser significativo para el cliente.
- Tener más de 10 miembros en el equipo requiere demasiada coordinación.
- Los Equipos muy grandes generan demasiada complejidad y dificultades en la comunicación como para que un proceso empírico pueda ser de utilidad.
- Los roles de Product Owner y Scrum Master no se cuentan como Developers, a menos que también estén contribuyendo al desarrollo del trabajo durante el Sprint.

---

1. El Product Owner (Dueño de Producto)

product-owner
Image not found or type unknown

- El Product Owner es el responsable de maximizar el valor entregado por los Developers
- Es el representante de todos los interesados (stakeholders) en los resultados del producto
- Actúa como interlocutor único ante el equipo Scrum, con autoridad para tomar decisiones.
- Es el encargado de garantizar los resultados del producto.
- El Product Owner (P.O) es una única persona, no un comité.
- Es considerado como “La voz del cliente”
- Es la única persona responsable de gestionar el Product Backlog.

La gestión del Product Backlog incluye:

- Expresar claramente los elementos del Product Backlog.
- Ordenar los elementos en el Product Backlog para alcanzar los objetivos y las misiones de la mejor manera posible.
- Garantizar que el Product Backlog sea visible, transparente y clara para todos y que muestre, lo que el equipo trabajará a continuación.
- Asegurar que los Developers entienden los elementos del Product Backlog a nivel necesario.
- El Product Owner es una única persona, no un comité. El Product Owner podría representar los deseos de un comité en el Product Backlog, pero aquellos que quieran cambiar la prioridad de un elemento del Product Backlog deben hacerlo a través del Product Owner.

Para que el Product Owner pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Product Owner se reflejan en el contenido y en la priorización del Product Backlog.

Responsabilidades del Product Owner

- Participar en las reuniones de planificación y revisión de las iteraciones (sprints).
- Entender las necesidades de las partes interesadas para posteriormente levantar los requerimientos que harán parte del Product Backlog.
- Mantener comunicación frecuente con el cliente (ya que es el único punto de contacto entre el cliente y el Equipo Scrum).
- Gestionar los riesgos globales del proyecto.
- Presentar informes del proyecto al cliente u otras partes interesadas.
- Aprobar cambios en el proyecto.

- Asegurar que se administran correctamente los recursos financieros del proyecto al inicio y durante su ejecución.
- Participar en las reuniones de apertura y cierre del proyecto.

Normalmente el rol del Product Owner se asocia con un Gerente de Proyecto, dadas sus responsabilidades de gestión, sin embargo, estos roles **NO son iguales.** Es más preciso asociarlo al rol que representa “La voz del cliente”.

---

2. Desarrolladores (Developers)

scrum-master

Los Developers son los profesionales que realizan el trabajo de entregar un Incremento de producto “Terminado” (Done) que potencialmente se pueda poner en producción al final de cada Sprint. Solo los Developers participan en la creación del Incremento.

Responsabilidades de los Developers

- Crear un plan para el Sprint: el Sprint Backlog
- Asegurar la calidad con el cumplimiento de la Definición de Terminado (Definition of Done)
- Adaptar su plan de trabajo diario para alcanzar el Objetivo del Sprint (Sprint Goal)
- Responsabilizarse del trabajo.

Nota: Se llaman “Desarrolladores”, porque desarrollan el producto (no necesariamente software)

La organización es la encargada de estructurar y empoderar a los Developers para que estos organicen y gestionen su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo Scrum en conjunto.

Los Developers tienen las siguientes características:

- **Son autogestionados.** Nadie (ni siquiera el Scrum Master) les da órdenes o imposiciones sobre cómo convertir elementos del Product Backlog en Incrementos de funcionalidad potencialmente desplegables.
- **Nadie fuera del Scrum Master y el Product Owner** puede pedir a los Developers que trabajen en un conjunto diferente de requisitos que previamente hayan sido planeados y acordados.
- **Los Developers son multifuncionales,** con todas las habilidades necesarias para crear un Incremento de producto.

- **Scrum no reconoce títulos para los miembros de un Equipo Scrum,** independientemente del trabajo que realice cada persona, quienes ejecuten el trabajo necesario para construir el producto se conocen como Developers.
- **No existen sub-equipos en el Equipo Scrum,** no importan los dominios particulares que requieran tenerse en cuenta, como pruebas, aseguramiento de calidad, arquitectura, operaciones, o análisis de negocio.
- **Los Developers pueden tener habilidades especializadas** y áreas en las que estén más enfocados, pero la responsabilidad del producto recae sobre los Developers como un todo.

---

3. El Scrum Master (Maestro Scrum)

scrum-master
Image not found or type unknown

El Scrum Master es un facilitador, responsable de promover y apoyar Scrum como se define en la Guía de Scrum. Su principal responsabilidad entonces es garantizar que todos conocen y aplican correctamente la teoría y práctica de Scrum.

- El Scrum Master es un líder que sirve al Equipo Scrum y en general a toda la organización.
- El Scrum Master ayuda a las personas externas al Equipo Scrum a entender qué interacciones con el Equipo Scrum pueden ser útiles y cuáles no.

El Scrum Master al servicio del Equipo y los Desarrolladores (Developers)

El Scrum Master sirve a los desarrolladores de varias formas, incluyendo:

- Guiar a los Desarrolladores para que aprendan a ser autogestionados y multifuncionales.
- Ayudar a los Desarrolladores a centrarse en crear productos de alto valor además de cumplir con la Definición de Terminado.
- Eliminar impedimentos para el progreso de los Desarrolladores.
- Facilitar los eventos de Scrum según se requiera o necesite, para que sean productivos, respetando el tiempo designado para cada uno (time-box).
- Guiar a los Desarrolladores en entornos organizacionales en los que Scrum aún no haya sido adoptado y entendido por completo.

El Scrum Master al servicio del Product Owner

El Scrum Master sirve al Product Owner de varias formas, incluyendo:

- Asegurar que los objetivos, el alcance y el dominio del producto sean entendidos por todos en el Equipo de la mejor manera posible.
- Ayudar a encontrar técnicas para gestionar el Product Backlog, definir el Objetivo de Producto (Product Goal) y gestionar los atrasos en el producto.
- Ayudar al Equipo Scrum a entender la necesidad de contar con elementos del Product Backlog claros y concisos.
- Entender la planificación del producto en un entorno empírico y complejo.
- Ayudar a entender y practicar la agilidad.
- Facilitar la colaboración de las partes interesadas y/o cliente según sea necesario.

El Scrum Master al servicio de la Organización

El Scrum Master sirve a la organización de varias formas, incluyendo:

- Liderar y guiar a la organización en la adopción de Scrum.
- Planificar y asesorar en la adopción de Scrum en la organización.
- Ayudar a los empleados e interesados a entender y poner en práctica un enfoque empírico en el desarrollo del trabajo.
- Motivar cambios que incrementen la productividad en la organización.
- Junto con otros Scrum Masters, incrementar la efectividad de la adopción de Scrum en la organización.
- Eliminar las barreras de comunicación entre los Equipos Scrum y las diferentes partes interesadas.

> Scrum **NO** considera el rol tradicional de “Gerente del Proyecto”. *Entonces, ¿a dónde van estos empleados?* Lo más frecuente y natural es que los Gerentes de Proyecto hagan una transición hacia Product Owners e incluso a Scrum Masters. Sin embargo, muchos buenos Scrum Masters provienen de los desarrolladores y QA; por lo tanto el Scrum Master no es ni remotamente parecido al del Gerente de Proyecto, y por esto es importante que el candidato a Scrum Master adquiera capacitación en Scrum.
>
> El Scrum Master debe aceptar que el equipo es dueño de la agenda de trabajo, así que no más actualizar los viejos Diagramas de Gantt. Debe alentar al equipo para que mantenga actualizadas sus estimaciones diarias. El Scrum Master asegura que se siga el proceso de Scrum, que todos entienden Scrum y cómo funciona.
>
> El Scrum Master debe quitar impedimentos, o asistir en quitar impedimentos, lo más rápido posible.
>
> Se asegura que el equipo se mantiene en rumbo, recordándole al equipo el objetivo del Sprint en cada reunión.

> Organiza los Scrums diarios (Reunión diaria), y se asegura que se realice en el mismo lugar a la misma hora.
>
> **Traducido de Coping with change on Scrum projects, por Jack Milunksy.**

Comunicación en el equipo

comunicacion-equipo
Image not found or type unknown

2.2 - Artefactos en Scrum

Los artefactos de Scrum representan el trabajo o el valor en diversas formas, los cuales son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento sobre el Producto.

Cada artefacto está vinculado con un compromiso para garantizar que se cuenta con información suficiente sobre el producto, esto mejora la transparencia y el enfoque con el que se puede medir el progreso:

- Para el **Product Backlog** el compromiso es el **Objetivo de Producto (Product Goal).**
- Para el **Sprint Backlog** el compromiso es el **Objetivo del Sprint (Sprint Goal).**
- Para el **Incremento** el compromiso es la **Definición de Terminado (Definition of Done - DoD).**

Antes de abordar los artefactos, ¿qué es un Producto?

Un producto es un vehículo para aportar valor y se refiere a cualquier resultado entregable y potencialmente utilizable que se desarrolla durante un determinado tiempo. Tiene un límite claro, partes interesadas conocidas, usuarios o clientes bien definidos. Un "producto" puede ser un bien tangible o intangible, como un software, una aplicación, un documento, un servicio, un producto físico o cualquier otro artefacto que proporcione valor a los clientes o usuarios finales.

---

1. El Product Backlog

El Product Backlog es una **lista ordenada de todos los elementos que podrían ser necesarios para el producto** y es la **única fuente de requisitos** para cualquier cambio a realizarse en el producto, y también es la única fuente de trabajo para el Equipo Scrum.

El Product Owner es el responsable del Product Backlog, incluyendo su contenido, disponibilidad y ordenación.

Un Product Backlog siempre esta en constante actualización. El desarrollo inicial de los elementos del Product Backlog solo refleja los requisitos conocidos y mejor entendidos al principio.

El Product Backlog evoluciona a medida que el producto y el entorno en el que se usará también lo hacen. El Product Backlog es dinámico; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. Mientras el producto exista, su Product Backlog también existe.

El Product Backlog enumera todas las características, funcionalidades, requisitos, mejoras, correcciones y cambios a realizarse sobre el producto para entregas futuras. Los elementos del Product Backlog tienen como mínimo los siguientes atributos:

- La descripción (en función del valor para el negocio)
- El orden (prioridad)
- La estimación
- Los criterios de aceptación

A medida que un producto es utilizado y se incrementa su valor y el mercado proporciona retroalimentación, el Product Backlog se convierte en una lista más larga y exhaustiva. Los elementos estan en constante actualización así que el Product Backlog es un artefacto vivo. Los cambios en los requisitos de negocio, las condiciones del mercado o la tecnología podrían causar cambios en el Product Backlog.

¡Nota Importante! Para evitar proyectos que nunca terminan, es importante considerar el **alcance del producto** a la hora de crear y actualizar el Product Backlog.

> **El compromiso del Product Backlog es el Product Goal**

Product Goal (Objetivo del Producto)

- Es el **estado futuro del producto** que el equipo toma como referencia para **planificar su trabajo.**
- El Product Goal podría incluir también las **restricciones** a las cuales se enfrenta el desarrollo del producto **(tiempo, costo, alcance, calidad)** con el fin de asegurar que se cumple con las necesidades del cliente.
- El Product Goal es el objetivo a largo plazo para el Equipo Scrum. Deben cumplir (o abandonar) un objetivo antes de asumir el siguiente.
- El Product Goal se encuentra en el Product Backlog.

---

2. El Sprint Backlog

El Sprint Backlog es un **conjunto de los elementos del Product Backlog que han sido seleccionados para desarrollarse durante un Sprint**, el cual constituye un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint.

El Sprint Backlog es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”.

Cuando se requiere nuevo trabajo, el Equipo de Desarrollo lo adiciona al Sprint Backlog. A medida que el trabajo se ejecuta o se completa se va actualizando la estimación de trabajo restante. Cuando algún elemento del plan se considera innecesario, es eliminado. Solo el Equipo de Desarrollo puede cambiar su Sprint Backlog durante un Sprint. El Sprint Backlog es una imagen visible en tiempo real del trabajo que el Equipo de Desarrollo planea llevar a cabo durante el Sprint y pertenece únicamente al Equipo de Desarrollo.

- El Sprint Backlog hace visible todo el trabajo que el Equipo de Desarrollo debe completar para alcanzar el Objetivo del Sprint.
- Para asegurar el mejoramiento continuo, en el Sprint Backlog se incluye por lo menos una actividad que permita la mejora de procesos (normalmente identificadas en la Retrospectiva inmediatamente anterior).
- El Sprint Backlog es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Scrum Diario.

El Sprint Backlog se compone del **Sprint Goal (por qué)**, el conjunto de **elementos del Product Backlog seleccionados para el Sprint (qué)**, así como un **plan de trabajo para entregar el Incremento (cómo).**

> **El compromiso del Sprint Backlog es el Sprint Goal**

Sprint Goal (Objetivo del Sprint)

Es el Objetivo único para el Sprint, el cual se verá reflejado al entregar el incremento de producto generado al final de un Sprint.

El objetivo de Sprint se define **durante la Planificación de Sprint** y, a continuación, se agrega al Backlog de Sprint (Sprint Backlog).

A medida que los desarrolladores trabajan durante el Sprint, tienen en cuenta el Objetivo del Sprint. Si el trabajo resulta ser diferente de lo que esperaban, colaboran con el Product Owner para

negociar el alcance del Sprint Backlog dentro del Sprint sin afectar el Objetivo del Sprint.

---

3. Incremento de Producto

Un Incremento es el resultado que entregan los Desarrolladores al final de cada Sprint, el cual se suma al valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint el nuevo Incremento debe estar **“Terminado”**, lo cual significa que está en condiciones de ser utilizado y que cumple con la Definición de “Terminado” del Equipo Scrum.

- El incremento es un paso hacia la visión del producto (Product Goal).
- El incremento debe estar en condiciones de utilizarse sin importar si el Product Owner decide liberarlo o no.

> **El compromiso del Incremento es la Definición de Terminado (DoD)**

Definición de Terminado (DoD - Definition Of Done)

Es una descripción formal del estado del Incremento cuando cumple con las medidas de calidad requeridas para el producto.

- La Definición de terminado (Criterios de terminado) es definida en conjunto por el Equipo Scrum.
- La DoD puede ser definida para cada incremento o para el producto en general.
- Si un incremento no cumple con la Definición de Terminado, no puede ser aprobado.

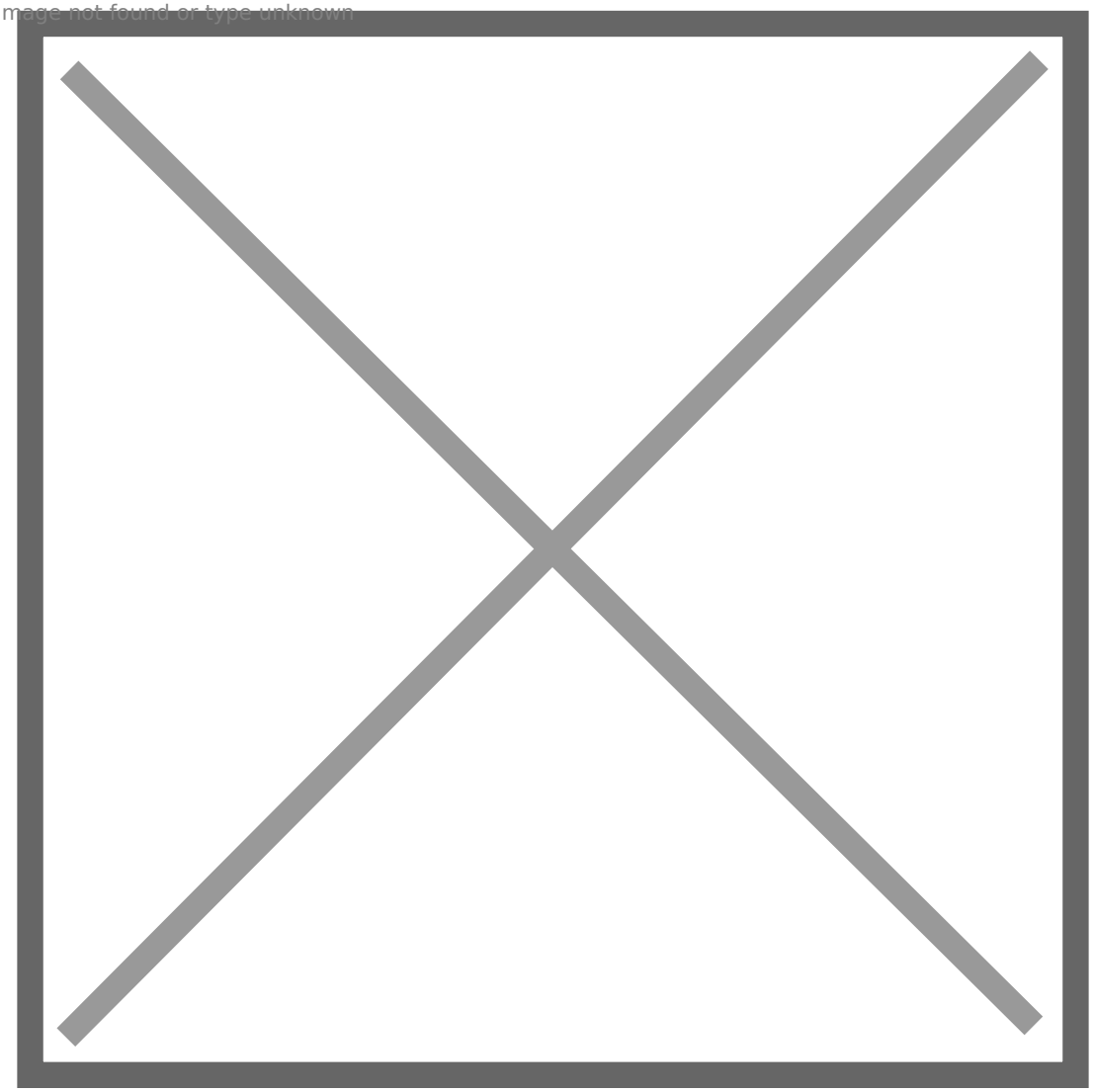
Un ejemplo **genérico** de criterios de terminado puede ser:

- *Los elementos fueron revisados por otros miembros del equipo (se realizaron las pruebas de colegas).*
- *No existen defectos o los defectos están arreglados.*
- *Se completaron las pruebas unitarias.*
- *Se finalizó toda la documentación técnica y de usuario final.*
- *Todos los archivos y documentos del proyecto están en el repositorio de la organización.*

2.3 - Eventos en Scrum

Los eventos en Scrum, también conocidos como sesiones de trabajo, son momentos estructurados que facilitan la colaboración, la transparencia y la inspección regular, promoviendo una ejecución ágil de los proyectos. Aquí está una descripción concisa de cada uno y su propósito:

Image not found or type unknown



- **Sprint:** Es el corazón de Scrum, un periodo de tiempo determinado (usualmente de dos a cuatro semanas) donde se realiza un incremento de producto "potencialmente entregable". Sirve para mantener al equipo enfocado en un conjunto de objetivos específicos.
- **Planificación del Sprint (Sprint Planning):** Al inicio de cada sprint, el equipo se reúne para planificar y acordar los elementos del backlog del producto que serán abordados durante ese sprint. Facilita la colaboración y la alineación de expectativas entre los miembros del equipo.

- **Daily Scrum (Scrum Diario):** Es una reunión corta (15 minutos) que tiene lugar cada día durante el sprint. Permite a los desarrolladores compartir actualizaciones sobre su progreso y planificar el trabajo del día, ayudando a identificar obstáculos tempranamente.
- **Revisión del Sprint (Sprint Review):** Al final del sprint, el equipo se reúne para evaluar y demostrar el trabajo realizado. Facilita la inspección del incremento de producto y ajustes basados en el feedback de los stakeholders.
- **Retrospectiva del Sprint (Sprint Retrospective):** Es una reunión que ocurre después de la revisión del sprint y antes de la próxima planificación del sprint. Sirve para que el equipo reflexione sobre su forma de trabajar y acuerde maneras de mejorar para el próximo sprint.

En Scrum existen diferentes eventos predefinidos con el fin de crear regularidad, minimizar la necesidad de reuniones innecesarias y permitir la transparencia.

Todos los eventos tienen un periodo de **tiempo limitado** (time-box), de tal modo que todos tienen una duración máxima, y se llevan a cabo al mismo tiempo y en el mismo lugar para reducir la complejidad.

1. El Sprint

¿De qué se trata un Sprint?

Los Sprints son el latido del corazón de Scrum, donde todas las ideas se convierten en valor. El Sprint es un evento con un tiempo asignado (time-box) de 1 a 4 semanas durante el cual se crea un incremento de producto “Terminado” utilizable y potencialmente desplegable.

Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior. Los Sprints contienen y consisten en la Planificación del Sprint, los Scrum Diarios, el desarrollo del trabajo, la Revisión del Sprint, y la Retrospectiva del Sprint.

Cada Sprint puede considerarse como un “mini-proyecto” con un horizonte no mayor de un mes. Al igual que los proyectos, los Sprints se usan para alcanzar una meta. Cada Sprint tiene un objetivo de lo que se construirá, es decir, el Objetivo del Sprint (Sprint Goal), un diseño y un plan flexible que guiará su construcción, el trabajo del equipo y el incremento de producto resultante.

Los Sprints habilitan la predictibilidad al asegurar la inspección y adaptación del progreso al menos en cada mes calendario.

¿Cuál es la duración de un Sprint?

Típicamente, el Sprint dura entre 1 y 4 semanas (un mes calendario), siendo 2 semanas el duración más común.

> Cuando el horizonte de un Sprint es mayor a un mes, la complejidad podría incrementarse y el riesgo podría aumentar.

¿Quiénes participan durante el desarrollo del Sprint y cómo se involucran?

- **Scrum Master:** Ayuda al equipo a mantenerse enfocado en la meta del Sprint, facilita la resolución de impedimentos y garantiza que se sigan las prácticas Scrum.
- **Product Owner:** Asegura que el equipo tenga una visión clara del valor que deben entregar, prioriza las tareas y proporciona claridad sobre las historias de usuario o requerimientos.
- **Developers:** Implementan las funcionalidades, mantienen una comunicación constante sobre el progreso y colaboran para cumplir con la meta del Sprint.

¿Qué se necesita para poder llevar a cabo un Sprint?

- Un Product Backlog priorizado.
- Un Sprint Goal o meta definido en la Planificación del Sprint.
- Un Sprint Backlog con los elementos seleccionados y su plan de trabajo.

¿Qué sucede durante el Sprint?

No hay una "agenda" para el Sprint como tal, pero el flujo general es:

1. Comenzar a trabajar en los elementos del Sprint Backlog.
2. Realizar reuniones diarias de sincronización (Daily Scrum).
3. Trabajar colaborativamente para resolver impedimentos.
4. Asegurarse de que se cumplan las definiciones de terminado ("Definition of Done") para los ítems completados.

¿Qué resulta del Sprint?

- Un incremento potencialmente entregable del producto.
- Actualizaciones al Product Backlog basadas en el aprendizaje y el trabajo durante el Sprint.

Consideraciones sobre la duración del Sprint

- Una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse.
- Durante el Sprint **no** se realizan cambios que puedan afectar al objetivo del Sprint.
- La calidad del producto no debe disminuir.
- El alcance puede clarificarse y renegociarse con el Product Owner a medida que se va aprendiendo más.

- Todos los eventos relacionados con el Sprint deben ejecutarse dentro del tiempo del Sprint (Planificación, Scrum Diario, Revisión y Retrospectiva), la falta de alguno de estos eventos da como resultado una reducción de la transparencia y constituye una oportunidad perdida de inspección y adaptación.
- La duración de los Sprint dependerá de los siguientes factores:
  - Fechas de entrega acordadas con el cliente
  - Experiencia del equipo (a menor experiencia con Scrum, los Sprints deberían ser más cortos)
  - Restricciones propias para la construcción del producto
- La duración del Sprint es definida por todo el Equipo Scrum con las recomendaciones del Scrum Master, quienes a su vez deberán llegar a un consenso con el Product Owner.

> **¿Se puede cancelar un Sprint?** Un Sprint puede cancelarse antes que el periodo de tiempo llegue a su fin. Algunas de las razones por las que se podría cancelar un Sprint son:
>
> - La organización cambia el Objetivo del Producto (Product Goal) y eso afecta el entregable del Sprint que se está ejecutando.
> - Las condiciones del mercado o de la tecnología cambian.
> - El objetivo del Sprint quedó obsoleto.
> - Surge un error sobre un producto que está en ambiente de producción y los Developers deben resolverlo cuanto antes.
> - Surge un cambio urgente que debe introducirse de inmediato al incremento de producto.

> **Consideraciones sobre la cancelación del Sprint**
>
> - **Solo el Product Owner tiene la autoridad para cancelar el Sprint,** aunque puede hacerlo bajo la influencia de los interesados, de los Developers o del Scrum Master.
> - Debido a la corta duración de los Sprints, **su cancelación rara vez tiene sentido.**
> - Cuando se cancela un Sprint **se revisan los elementos que se hayan completado y "Terminado".** Si una parte del trabajo es potencialmente entregable, el Product Owner normalmente la acepta.
> - Si el Sprint ha sido cancelado, todos los elementos del Sprint Backlog que no hayan sido completados se vuelven a estimar para el siguiente Sprint o se vuelven a introducir en el Product Backlog para desarrollarse en futuros Sprints.
> - Aunque se cancele un Sprint de igual manera consume recursos ya que el equipo debe reagruparse de nuevo para otra Planificación de Sprint para empezar otro Sprint.

> - Las cancelaciones del Sprint son a menudo traumáticas para el Equipo Scrum y son muy poco comunes.

---

2. Planificación del Sprint (Sprint Planning)

¿De qué se trata la Planificación del Sprint?

La Planificación del Sprint es una sesión de trabajo en la que el equipo Scrum determina qué elementos del Product Backlog se trabajarán durante el próximo Sprint y cómo se abordarán, es decir un plan de trabajo, y este plan se crea mediante el trabajo colaborativo de todo el equipo Scrum.

En esta sesión se establece el alcance y el objetivo para las próximas semanas (Sprint Goal).

- El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito.
- El Scrum Master enseña al Equipo Scrum a mantenerse dentro del tiempo máximo del evento.
- El Product Owner es el encargado de dirigir la sesión de planificación, ya que él tiene la información que se usará como insumo para seleccionar el trabajo.

¿Cuál es la duración de la Planificación del Sprint?

Máximo 8 horas para un Sprint de un mes. Para Sprints más cortos, la duración se reduce proporcionalmente (por ejemplo, 4 horas para un Sprint de dos semanas).

¿Quiénes participan de la Planificación del Sprint y cómo se involucran?

- **Scrum Master:** Facilita la sesión y asegura que todos entiendan el propósito y la agenda de la sesión. Resuelve impedimentos que puedan surgir durante la planificación.
- **Product Owner:** Explica claramente los ítems del Product Backlog y ayuda al equipo a priorizar el trabajo basándose en el valor y las necesidades del negocio. Propone el objetivo del Sprint.
- **Developers:** Determinan el trabajo que pueden realizar durante el Sprint y descomponen los ítems en tareas con sus respectivas estimaciones.

¿Qué se necesita para la Planificación del Sprint?

- El Product Backlog actualizado.
- El último incremento del producto.
- Rendimiento pasado del equipo (por ejemplo, velocidad).
- "Borrador" de los elementos opcionados para trabajar en este Sprint (identificados previamente por el Product Owner).

Agenda de la Planificación del Sprint

1. ¿Por qué este Sprint es valioso?

- El Product Owner propone cómo el producto podría aumentar su valor y utilidad en el Sprint.
- Todo el equipo de Scrum colabora para definir el Objetivo de Sprint (Sprint Goal) durante la Planificación.

2. ¿Qué puede hacerse en este Sprint?

- Se identifican los cambios y/o riesgos que puedan afectar este Sprint.
- En equipo se seleccionan los elementos del Product Backlog que harán parte de este Sprint, es decir, definen el Sprint Backlog.
- Se identifican las dependencias que puedan existir.
- Los elementos seleccionados se descomponen en tareas detalladas y se realiza su estimación.
- Los Developers (y algunos casos también el Scrum Master si ejecuta trabajo del Sprint) se autoasignan los elementos del Sprint Backlog.

3. ¿Cómo se conseguirá completar el trabajo seleccionado?

- Se explica el detalle de las tareas y esfuerzos necesarios que deben ejecutarse para "terminar" los elementos del Sprint Backlog.
- Se asegura que todo el Equipo entiende cómo crear un incremento cumpliendo con la Definición de Terminado (Definition Of Done).

¿Qué resulta de la Planificación del Sprint?

- El objetivo del Sprint (Sprint Goal) definido.
- Lista de ítems del Product Backlog seleccionados para el Sprint (Sprint Backlog).
- El plan sobre cómo el equipo abordará las tareas (como parte del Sprint Backlog).

Consideraciones adicionales sobre la Planificación del Sprint

- La reunión debe ser colaborativa; todas las decisiones se toman en conjunto.

- Es vital no sobrecargar al equipo durante el Sprint. Es preferible subestimar que sobreestimar.
- La claridad en el objetivo del Sprint ayuda a mantener al equipo alineado y enfocado.
- El equipo debe estar preparado para adaptarse a los cambios, manteniendo la visión y el objetivo del Sprint.

---

3. Daily Scrum (Scrum Diario)

¿De qué se tratan los daily?

El Scrum Diario es una corta sesión diaria (15 minutos) que tiene como objetivo inspeccionar el progreso de los Developers hacia el Objetivo del Sprint (Sprint Goal) e identificando posibles impedimentos para la ejecución.

El Scrum Diario se realiza para cada día del sprint.

Nota: Si el Product Owner o el Scrum Master están desarrollando elementos del Sprint Backlog, entonces pueden participar como Developers.

¿Cuál es la duración de los daily?

Máximo 15 minutos.

¿Quiénes participan de los daily y cómo se involucran?

- **Scrum Master:** Asegura que la reunión ocurra diariamente, facilita la solución de impedimentos externos y mantiene la estructura y duración de la reunión.
- **Product Owner:** Puede asistir para escuchar, pero no interfiere en la dinámica de la reunión. Si tiene comentarios o inquietudes, los discute fuera del Daily Scrum.
- **Developers:** Activamente participan compartiendo su progreso, plan para el día y señalando impedimentos.

Nota: El Scrum Master y Product Owner pueden asistir, pero la reunión es principalmente para los Developers.

¿Qué se necesita para los daily?

- Sprint Backlog.
- Progreso del día anterior (tareas completadas, en progreso, en pruebas, pendientes).

- Posibles impedimentos identificados.

Posible agenda de los daily

Nota: El Scrum Master y los Developers tienen la libertad de seleccionar la estructura y técnicas que deseen para la ejecución de esta reunión; sin embargo, el Scrum Master tiene la **opción** de hacer las siguientes preguntas a los Developers:

- ¿Qué hice hoy?
- ¿Qué haré mañana?
- ¿Existe algún impedimento que esté afectando o nos pueda afectar?

¿Qué resulta de los daily?

- Entendimiento compartido del progreso hacia el Objetivo del Sprint.
- Identificación y registro de impedimentos.
- Ajustes en el plan diario de trabajo, si es necesario.

Consideraciones adicionales sobre los daily

- El Scrum diario se realiza a la misma hora y lugar todos los días para reducir la complejidad. Por lo general se realiza antes de finalizar la jornada de trabajo del Equipo.
- Durante el Scrum Diario el Equipo coordina y planea el trabajo para las siguientes 24 horas.
- Si surgen temas que requieran discusiones más amplias, se deben tratar fuera del Scrum diario para no desviar el propósito principal.
- El Scrum Master se asegura de que la reunión se ejecute diariamente, pero son los Developers los responsables de dirigir el Scrum Diario, es decir que, no es obligatoria la presencia del Scrum Master para que se lleve a cabo el Scrum Diario.
- El Scrum Master enseña al Equipo a mantener el Scrum Diario dentro de un bloque de tiempo de máximo 15 minutos.
- El Scrum Diario es una reunión interna del Equipo. Si otras personas están presentes, el Scrum Master se asegura de que no interrumpan la reunión ni alteren la planificación del trabajo.
- Los Scrum Diarios mejoran la comunicación, identifican los impedimentos relacionados con desarrollo del trabajo, resaltan y promueven la toma rápida de decisiones y mejoran el nivel de conocimiento del Equipo.
- Durante los Scrum Diarios se actualizan el Burndown chart y el Scrum Board.
- Esta reunión por lo general se realiza de pie con el fin de garantizar el cumplimiento del tiempo asignado, por lo que también se le puede conocer como el Scrum Diario de Pie (Daily Standup Meeting).

---

4. Revisión del Sprint (Sprint Review)

¿De qué se trata la Revisión del Sprint?

Al final del Sprint se lleva a cabo la Revisión de Sprint en la que se inspecciona el Incremento que se produjo durante el Sprint y adaptar el Product Backlog si fuese necesario.

El Equipo Scrum realiza una demostración del incremento y se revisa el progreso con respecto al Objetivo de Producto; basándose en esto y en cualquier cambio que deba considerarse, los asistentes colaboran para determinar los elementos que podrían desarrollarse en el siguiente Sprint para optimizar el valor.

Se trata de una sesión de trabajo, no de una reunión de seguimiento, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación para el equipo Scrum y fomentar la colaboración.

El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado.

¿Cuál es la duración de la Revisión del Sprint?

Normalmente, dura **máximo 4 horas para un Sprint de 4 semanas** (la duración suele ser proporcional al largo del Sprint, por lo que para Sprints de 2 semanas, suele ser de 2 horas).

¿Quiénes participan de la Revisión del Sprint?

- **Scrum Master:** Facilita la sesión y garantiza que se siga la agenda.
- **Product Owner:** Presenta los ítems del Product Backlog completados, resalta el valor del trabajo realizado y lidera la discusión sobre las prioridades.
- **Developers:** Demuestran el trabajo realizado y responden preguntas técnicas.
- **Stakeholders:** Brindan retroalimentación y aclaran dudas o expectativas.

¿Qué se necesita para la Revisión del Sprint?

- Incremento de producto terminado del Sprint.
- Product Backlog.
- Estimaciones del trabajo (en el Sprint Backlog).

Agenda de la Revisión del Sprint

- El Equipo habla acerca de qué estuvo bien durante el Sprint, qué problemas aparecieron y cómo fueron resueltos esos problemas.
- El Equipo hace una demostración del trabajo que ha "Terminado" y responde preguntas acerca del Incremento y su funcionalidad.
- El Product Owner habla acerca del Product Backlog en su estado actual. Proyecta objetivos probables y posibles fechas de entrega basándose en el progreso obtenido hasta la fecha (si fuera necesario).
- El Product Owner realiza la aprobación de los elementos del Sprint Backlog que se han "Terminado" y rechaza los que no se han "Terminado" y con esto determinar la Deuda Técnica.
- Seguimiento y control del presupuesto, capacidades potenciales y mercado para la próxima entrega prevista del producto.
- Revisión del estado de los cambios que fueron planificados para este Sprint y revisión del estado de los riesgos que afectaron este Sprint.
- Refinamiento (grooming) del Product Backlog, en el que todo el Equipo colabora acerca de qué hacer a continuación, de modo que esta reunión proporcione información de entrada valiosa para Reuniones de Planificación de Sprints subsiguientes.

¿Qué resulta de la Revisión del Sprint?

- El resultado de la Revisión de Sprint es un Sprint Backlog revisado, además de una identificación de alto nivel de los posibles elementos del Product Backlog que se pueden desarrollar para el siguiente Sprint.
- Retroalimentación sobre el Incremento de producto.
- Actualización del Product Backlog con posibles nuevos ítems o cambios.
- Claridad sobre la dirección y prioridades del producto.

Consideraciones adicionales la Revisión del Sprint

- Aunque se busca retroalimentación, la Revisión del Sprint no es únicamente una sesión de aprobación; se verifica que el trabajo esté "terminado" según la Definición de Terminado (DoD).
- Es esencial tener un ambiente abierto donde las partes interesadas sientan la libertad de dar retroalimentación honesta y constructiva sobre el producto.

---

5. Retrospectiva del Sprint (Sprint Retrospective)

¿De qué se trata una retrospectiva?

La Retrospectiva del Sprint es una sesión donde el equipo Scrum reflexiona sobre el Sprint que acaba de terminar para identificar oportunidades de mejora y planificar acciones concretas para próximos Sprints.

¿Cuál es la duración de una retrospectiva?

Suele durar alrededor de 3 horas para un Sprint de 4 semanas (puede ser proporcionalmente menos para Sprints más cortos, por ejemplo, alrededor de 45 minutos a 1.5 horas para Sprints de 2 semanas).

¿Quiénes participan y cómo se involucran?

- **Scrum Master:** Facilita la discusión, asegura un ambiente seguro para que todos expresen sus opiniones y ayuda al equipo a identificar áreas de mejora.
- **Product Owner:** Aporta desde su perspectiva sobre el desarrollo del producto, prioridades y feedback del cliente.
- **Developers:** Comparten feedback, discuten desafíos y proponen soluciones.

¿Qué se necesita para el evento?

- Feedback y observaciones del equipo sobre el Sprint que terminó.
- Datos y métricas relevantes del Sprint (como el Burndown Chart).
- Acciones de mejora de Retrospectivas anteriores.

Agenda de una retrospectiva

- Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas.
- Reflexionar sobre lo que salió bien y lo que no durante el Sprint.
- Discutir las causas de los desafíos y problemas.
- Identificar oportunidades de mejora y priorizarlas.
- Crear un plan o establecer acciones concretas para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo.

¿Qué resulta de una retrospectiva?

- Acciones concretas para implementar mejoras en el próximo Sprint.
- Comprensión compartida de cómo mejorar el proceso de trabajo.

Consideraciones adicionales de una retrospectiva

- La Retrospectiva es un espacio seguro donde todos los miembros del equipo deben sentirse cómodos compartiendo su feedback sin temor a represalias.
- Las acciones de mejora deben ser específicas, medibles y asignadas para garantizar su implementación.

- Es esencial revisar las acciones de mejora de retrospectivas anteriores para evaluar su impacto y asegurar la mejora continua.

2.4 - Teoría de Scrum

- Scrum se basa en el empirismo (experiencia) y el pensamiento lean (reducir desperdicios).
- Scrum emplea un enfoque iterativo e incremental para optimizar la previsibilidad y controlar el riesgo.
- Scrum combina cuatro eventos formales para inspección y adaptación dentro de un contenedor: el Sprint.
- Los eventos funcionan porque se basan en los pilares empíricos de Scrum de **transparencia, inspección y adaptación.**

Transparencia

El proceso y el trabajo deben ser visibles para aquellos que realizan el trabajo (equipo Scrum), así como para los que reciben el trabajo (cliente y partes interesadas). Con Scrum, las decisiones importantes se basan en el estado percibido de los artefactos. Los artefactos que tienen poca transparencia pueden conducir a decisiones que disminuyen el valor y aumentan el riesgo. La transparencia permite la inspección.

Inspección

Los artefactos de Scrum y el progreso hacia los objetivos deben ser inspeccionados con frecuencia y diligentemente para detectar varianzas o problemas potencialmente indeseables. Para ayudar con la inspección, Scrum proporciona cadencia en forma de sus cinco eventos.

En los proyectos Scrum se deben inspeccionar frecuentemente los artefactos y el progreso hacia un objetivo, para detectar variaciones.

La inspección no debe ser tan frecuente como para que interfiera en el trabajo, las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo (normalmente el Scrum Master).

Adaptación

Si algún aspecto de un proceso se desvía fuera de los límites aceptables o si el producto resultante es inaceptable, el proceso que se está aplicando o los resultados que se producen deben ajustarse. El ajuste debe realizarse lo antes posible para minimizar la desviación adicional.

Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables, y que el producto resultante no será aceptable, el proceso o el material que está siendo procesado deben ser ajustados. Dicho ajuste debe realizarse cuanto antes para minimizar desviaciones mayores.

Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación, tal y como se describen en la sección Eventos de Scrum de esta guía:

- Reunión de Planificación del Sprint.
- Scrum Diario.
- Revisión del Sprint.
- Retrospectiva del Sprint.

teoria scrum

> La teoría de Scrum ha sido tomada de: https://scrumguides.org/

3. Scrum Master: Líder al servicio del equipo Scrum

En este capítulo, nos adentraremos en la función vital que desempeña el Scrum Master en la implementación de Scrum. El Scrum Master no solo asegura una adopción precisa de Scrum según lo delineado en la Guía de Scrum, sino que también facilita la comprensión y el respeto de su teoría y prácticas a todos los niveles organizacionales. A lo largo de este segmento, desplegaremos un análisis detallado de las diversas actividades, habilidades y responsabilidades que caracterizan esta función, ampliando lo ya explorado en el modelo core de Scrum. Puntos Clave de este Capítulo - Profundizar en la responsabilidad crítica del Scrum Master en la adopción y mejora de Scrum. - Analizar las actividades clave que el Scrum Master facilita dentro del equipo Scrum. - Estudiar las habilidades esenciales que un Scrum Master debe cultivar para liderar con eficacia. - Examinar las responsabilidades a nivel organizacional que incumben al Scrum Master.

3.1 - Habilidades del Scrum Master

Un Scrum Master debería tener un conocimiento profundo del marco Scrum por varias razones:

- Para poder aplicarlo de manera coherente y consistente
- Para poder resolver problemas y superar obstáculos
- Para poder adaptar Scrum a las necesidades del equipo
- Para poder enseñar y orientar a otros

Como scrum master (SM) es necesario brindar soporte y soluciones al Equipo Scrum y a la organización en general; para esto debe contar con aptitudes y actitudes que contribuyan al liderazgo, a continuación se muestran algunas habilidades que tiene un SM:

- Habilidades de liderazgo
- Habilidades de facilitación
- Habilidades de comunicación
- Habilidades para la resolución de problemas
- Habilidades de enseñanza

Habilidades de liderazgo

Las habilidades de liderazgo en un Scrum Master se refieren a la capacidad de liderar y guiar al equipo para que alcance sus metas y objetivos. Algunas de las habilidades de liderazgo esenciales para un Scrum Master incluyen:

- Fomentar la colaboración y el trabajo en equipo
- Ayudar a los miembros del equipo a superar obstáculos
- Ser una fuente de apoyo y orientación
- Establecer objetivos y expectativas claras
- Inspirar y motivar al equipo

> **Ejemplo:** Durante un proyecto, el equipo se siente inseguro respecto a la nueva tecnología que están utilizando. El Scrum Master, reconociendo la necesidad de liderazgo, organiza una serie de sesiones de capacitación con expertos para

> familiarizar al equipo con la tecnología, estableciendo así objetivos claros y proporcionando apoyo y orientación.

Tip

Recuerda que un buen líder no sólo dirige, sino también inspira y motiva. Ser un líder en Scrum significa servir al equipo, eliminando obstáculos y ayudando a cada miembro a dar lo mejor de sí.

Habilidades de facilitación

El SM logra que todas las acciones, procesos y tareas sean fáciles de entender para el equipo, además de asegurar que se cuente con el entorno y herramientas necesarias para trabajar. Es necesario que actúe de manera:

- Colaborativa: para guiar al equipo en sus funciones
- Neutral: para entender las necesidades de todas las partes sin tener preferencia alguna
- Responsable: garantizando la disciplina y la efectividad del equipo

> **Ejemplo:** Durante la retrospectiva, algunos miembros del equipo discuten sobre qué metodología de trabajo es la más eficiente. El Scrum Master, como facilitador, interviene para asegurar que todos tengan una oportunidad de expresarse y que la discusión se mantenga centrada en la mejora continua, evitando desviarse hacia debates improductivos.

El Scrum Master facilitando los eventos

El SM como facilitador debe tener en cuenta los siguientes aspectos al momento de moderar un evento:

- Gestionar un evento: es necesario que previamente se cuente con la información necesaria, tal como: objetivos, duración, participantes y demás.
- Asegurar las intervenciones necesarias: para que todos puedan participar y tomar el liderazgo.
- Controlar el ritmo: para que no se generen bucles o discusiones innecesarias, y se cumpla el objetivo del evento.

Habilidades de comunicación

Las habilidades de comunicación en un Scrum Master se refieren a la capacidad de comunicarse de manera efectiva con todas las personas con las que trabaja. Algunas de las habilidades de comunicación esenciales para un Scrum Master incluyen:

- Transmitir de manera clara y concisa los objetivos y expectativas del producto
- Brindar feedback constructivo
- Escuchar activamente
- Comunicar cambios y decisiones de manera clara
- Resolver conflictos

> **Ejemplo:** Uno de los desarrolladores, Juan, ha estado teniendo dificultades con una tarea específica. En vez de señalarlo abiertamente y ponerlo en una situación incómoda, el Scrum Master se acerca de manera privada, ofrece feedback constructivo sobre lo que ha observado y escucha activamente las preocupaciones de Juan, buscando soluciones conjuntas.

Tip

Recuerda que la comunicación no es solo hablar, sino también escuchar. Establece canales de comunicación abiertos, promueve la retroalimentación constante y verifica siempre que los mensajes hayan sido entendidos correctamente por todos los miembros del equipo.

Habilidades para la resolución de problemas

Las habilidades de resolución de problemas en un Scrum Master se refieren a la capacidad de identificar y resolver problemas que puedan surgir durante el trabajo con el equipo, esto considerando que No puedes impedir que aparezcan conflictos o problemas, pero puedes gestionarlos para minimizar el impacto al equipo, para esto debes desarrollar:

- Inteligencia emocional: individual y colectiva
- Entender las necesidades: como persona, más que como empleado y así abordar los conflictos oportunamente.
- Analizar la disposición de las personas y enfocarte en el problema.
- Hacer acuerdos para avanzar.
- Mantener una comunicación asertiva para superar el conflicto con madurez.

Tip

En lugar de brindar soluciones inmediatas, guía al equipo a través del proceso de descubrimiento de soluciones. Esto empodera al equipo y les ayuda a desarrollar habilidades de resolución de problemas por sí mismos.

El Scrum Master como Eliminador de impedimentos

> **Un impedimento es cualquier cosa que obstruye el flujo de trabajo generando retrasos.**

Es importante que el SM pueda remover los impedimentos del equipo para garantizar el flujo y a su vez el éxito del Sprint.

> **Ejemplo:** El equipo enfrenta un bloqueo técnico que ha detenido el progreso. El Scrum Master organiza una sesión de tormenta de ideas (brainstorming) donde todos pueden compartir posibles soluciones. Se anima al equipo a pensar de manera creativa, y juntos, logran encontrar un enfoque innovador que resuelve el problema.

Desafíos para el Scrum Master al eliminar impedimentos:

Desafío	¿Cómo lo puedes abordar?
El equipo Scrum no reporta el impedimento. Por lo tanto el SM desconoce la situación real del ambiente y no puede solucionar oportunamente.	- Preguntar para identificar falencias del equipo. - Trabajar en conjunto para solucionar - Fortalecer las habilidades de comunicación
El equipo se estanca con un impedimento fácil de resolver. El SM puede actuar oportunamente, hablando con el equipo sin afectar el flujo del proceso, brindando soluciones en tiempo real.	- Encontrar soluciones aplicables - Hacer una pausa para despejar la mente - Analizar el impedimento de manera creativa con el equipo
Falta capacidad y autoorganización en el equipo. El SM debe conocer la cultura y el contexto organizacional, procesos, proyectos y otros equipos de trabajo para apoyarse y buscar una solución al impedimento.	- Buscar personal idóneo para completar la capacidad del equipo - Apoyarse en otros equipos para encontrar la solución al impedimento

Habilidades de enseñanza

El SM debe animar al equipo para aprender nuevos conocimientos, esto permite que el equipo sea multidisciplinario y no se estanque fácilmente. Es necesario que el Scrum Master:

- Tenga seguridad para guiar al equipo
- Desarrollar comunicación abierta y transparente
- Establecer reglas claras con el equipo
- Involucrarse en todas las actividades (no como supervisor, sino como apoyo)
- Recordar que trabajas con personas no robots

Tip

Aprovecha los momentos informales para enseñar y compartir conocimientos. En ocasiones, una conversación casual puede ser una oportunidad dorada para impartir una lección valiosa. Recuerda, el aprendizaje no siempre tiene que ser estructurado o formal; a veces, los momentos espontáneos dejan las enseñanzas más duraderas.

El Scrum Master como coach y mentor

Como SM sueles entrenar a todo el equipo según sus habilidades y su estilo de trabajo.

Como coach: Su enfoque está en enseñar a los demás a mejorar. Un coach no necesariamente sabe más, busca ayudarte a sacar lo mejor de ti, para que encuentres mejores soluciones con tus propias habilidades.

Como mentor: Su enfoque está en ayudar y guiar a quienes tienen menos experiencia. Un mentor sí es alguien que sabe más que los demás sobre un determinado dominio y usa su conocimiento para guiar a los demás.

> **Ejemplo:** Sara, una nueva integrante del equipo, no está familiarizada con las prácticas de Scrum. El Scrum Master organiza sesiones individuales y grupales de capacitación para enseñarle los conceptos clave. A través de ejercicios prácticos y discusiones, el Scrum Master asegura que Sara comprenda no sólo los procesos sino también los valores y principios detrás de Scrum.

Habilidades complementarias

- **Habilidades de escucha:** Es importante resaltar la capacidad de escuchar, no sólo oír. Escuchar activamente implica comprender las preocupaciones, inquietudes y necesidades del equipo.
- **Capacidad de autoaprendizaje y adaptabilidad:** El mundo de la agilidad está en constante evolución. Un Scrum Master debe estar dispuesto a aprender y adaptarse a los cambios.
- **Integridad:** Es vital que un Scrum Master sea honesto y ético en todas sus acciones. Esto construye confianza y establece un ejemplo para el resto del equipo.
- **Empatía:** Poder ponerse en el lugar de los demás es fundamental para comprender las necesidades y preocupaciones del equipo.
- **Visión estratégica:** Aunque el SM no es el responsable del producto, es útil que entienda la visión a largo plazo para guiar mejor al equipo.
- **Gestión del tiempo:** Dado que los Sprints son ciclos de tiempo limitado, la habilidad de gestionar el tiempo eficientemente es crucial.

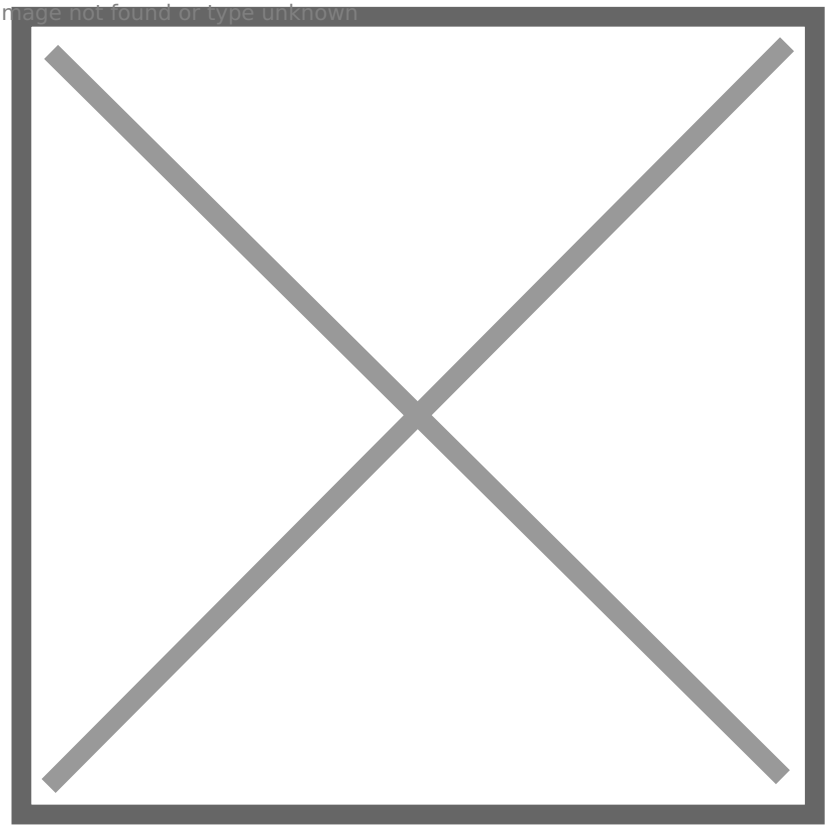
Tip

Como Scrum Master, predica con el ejemplo en cuanto a la gestión del tiempo. Utiliza herramientas de planificación, establece prioridades claras, aprende a decir "no" cuando sea necesario para asegurarte de que las tareas cruciales se cumplan dentro de los plazos. ¡Recuerda, trabajar inteligentemente es tan importante como trabajar arduamente!

3.2 - Los valores del equipo

Es importante que entre los miembros del equipo se genere y mantenga una filosofía basada en valores que fomenten la confianza, la comunicación y la entrega de resultados. A continuación, se relacionan los valores que se deben promover en un equipo Scrum:

Image not found or type unknown



- Compromiso
- Enfoque
- Apertura
- Respeto
- Coraje

Nota: El Scrum Master promueve los valores en el equipo.

En un equipo Scrum, los valores juegan un papel central en la construcción de una base sólida para la colaboración y la entrega efectiva.

- El **Compromiso** se refleja en la responsabilidad compartida, donde cada miembro se dedica plenamente a alcanzar los objetivos del equipo.

> **Ejemplo:** Tras la planificación de un Sprint, el equipo se da cuenta de que una de las historias de usuario comprometidas es más compleja de lo que inicialmente pensaban. En lugar de relegarla o dividirla, el equipo decide organizarse de forma diferente, redistribuyendo algunas tareas para asegurarse de que todos los elementos comprometidos se entreguen a tiempo.

- El **Enfoque** significa mantener una visión clara de las metas, priorizando tareas y asegurándose de que el esfuerzo se invierta en lo que realmente importa.

> **Ejemplo:** A mitad de un sprint, la dirección de la empresa sugiere una nueva funcionalidad que considera urgente. El equipo decide mantenerse enfocado en las tareas comprometidas para ese sprint y propone discutir la nueva solicitud en la próxima reunión de planificación, asegurando que la dirección actual del sprint no se vea comprometida.

- La **Apertura** implica una comunicación transparente y honesta, creando un ambiente donde las ideas y preocupaciones puedan ser compartidas libremente, y donde el feedback sea recibido como una oportunidad de mejora.

> **Ejemplo:** Al final de un sprint, una característica desarrollada no cumple con los estándares de calidad esperados. En lugar de intentar "maquillar" los resultados o posponer la revisión, el equipo comunica abiertamente la situación en la Revisión del Sprint, expone las causas del problema y propone una estrategia de corrección.

- El **Respeto** se manifiesta en el reconocimiento de las habilidades y opiniones de cada miembro, fomentando una cultura de empatía y consideración mutua, asegurando que cada voz se escuche y valorando la diversidad en el equipo.

> **Ejemplo:** Un miembro nuevo se une al equipo con una forma diferente de trabajar y con herramientas con las que el equipo no está familiarizado. En lugar de resistirse o desestimar su enfoque, el equipo respeta y se toma el tiempo para entender las ventajas que podría traer este nuevo método, valorando la diversidad de experiencia que el nuevo miembro aporta.

- Finalmente, el **Coraje** es esencial para enfrentar desafíos y cambios, permitiendo al equipo tomar riesgos calculados y aprender de los errores, y abordar situaciones difíciles con determinación.

>

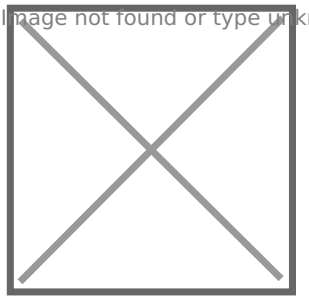
> **Ejemplo:** Después de varios días trabajando en una característica, el equipo se da cuenta de que la solución que estaban desarrollando no es óptima. Aunque revertir y cambiar de rumbo podría parecer desalentador, el equipo decide correr el riesgo, ya que creen que es lo mejor para el producto final. Hacen frente al desafío y trabajan juntos para encontrar una solución más adecuada.

Estos valores no sólo fortalecen la relación del equipo sino que también potencian su rendimiento, creando una cultura de excelencia. Cuando estos valores son asimilados por el equipo de Scrum (reforzados continuamente) y con las personas con las que trabajan, la teoría de Scrum que enfatiza en la transparencia, inspección y adaptación se materializa construyendo confianza.

3.3 - Los Pilares de Scrum

Si bien, los 12 principios del manifiesto ágil describen una orientación sobre los principios que se siguen en Scrum, hay 5 elementos que son clave para el buen funcionamiento de este marco de trabajo.

Estos pilares junto a los 12 principios del manifiesto ágil se deberían conocer y aplicar por los miembros del Equipo Scrum. El Scrum Master suele ser el responsable de que estos pilares hagan parte de la cultura del equipo.



1. Auto-gestión en el Equipo Scrum

Para Scrum, es altamente recomendable que el equipo tenga la capacidad de auto-gestionarse, esto significa que son los miembros del equipo quienes eligen la mejor opción de llevar a cabo su trabajo sin ser dirigidos por personas externas al equipo.

La auto-gestión es una característica definitoria de los equipos Scrum, centrada en la confianza, autonomía y responsabilidad de sus miembros. Aquí es donde los equipos no sólo toman decisiones sobre cómo abordar su trabajo, sino que también asumen la responsabilidad de esas decisiones.

A continuación, se listan algunos elementos clave que hace posible la Auto-Gestión del Equipo Scrum:

- Los desarrolladores deberían autoasignarse el trabajo a realizar en los diferentes Sprints, nadie, ni siquiera el Product Owner debe imponer el trabajo al Equipo.
- El equipo decide la mejor distribución del trabajo, garantizando la equidad y el trabajo en equipo.

- El Equipo debe conocer muy bien sus límites de decisión para así poder tener mayor autonomía (delegación).
- El Equipo debe tener espacios que le permitan realizar jornadas de investigación y capacitación.
- Se debe trabajar continuamente en la motivación del equipo.
- No se deben cambiar los miembros del equipo (a menos que sea inevitable).

□□ Tip para el Scrum Master

Las mejores prácticas y herramientas evolucionan con el tiempo. Dedica tiempo para la formación continua y asegúrate de que el equipo esté al día con las últimas tendencias y técnicas en Scrum.

Propiedad y responsabilidad colectiva del producto

Para garantizar una colaboración constante y evitar con el tiempo la aparición de una cultura de la culpa, es importante fomentar una propiedad colectiva del producto, esto significa que todo el Equipo Scrum es dueño del producto, y por tanto cualquiera de sus integrantes podría contribuir al desarrollo de cualquier parte del producto aun cuando no haya sido quien lo desarrolló inicialmente.

Así mismo no deberían existir reconocimientos individuales a los miembros del equipo por sus contribuciones al producto.

□□ Tip para el equipo

Evita la cultura de la culpa: En lugar de buscar a quién culpar, enfócate en cómo resolver el problema.

Colaboración de equipo

La colaboración se da gracias a la constante comunicación que existe en los Equipos Scrum, tanto entre sus miembros como con las Partes interesadas del Proyecto, este concepto es parte integral del Manifiesto Ágil **"La forma más eficiente y efectiva de transmitir información hacia y dentro del Equipo de Desarrollo es la conversación cara a cara"**.

El Scrum Master es el rol responsable de garantizar una sana comunicación entre todas las partes interesadas del proyecto, en especial de su Equipo de Desarrollo.

□□ Tip para el equipo

Fomentar una cultura de feedback constante puede simplificar muchos aspectos del trabajo. Las correcciones tempranas y la orientación adecuada evitan la complejidad de corregir errores más adelante.

Se debe considerar que, según la naturaleza del proyecto, las necesidades de la organización e incluso factores externos, se determinan la ubicación de los miembros del Equipo:

- **Equipos Centralizados:**
  - Los miembros del Equipo se encuentran en la misma ubicación, lo que les permite comunicarse con gran facilidad.

    - La resolución de problemas es prácticamente inmediata, ya que al estar ubicados en el mismo lugar es fácil realizar sesiones de diálogo.
- **Equipos Distribuidos:** Es aquel en el que sus miembros no se encuentran en una misma ubicación, por lo general está disperso debido a factores como:
    - La subcontratación (outsourcing – freelance).
    - Oficinas ubicadas en diferentes ubicaciones físicas.
    - Trabajo desde casa.

  Para garantizar la comunicación permanente en este tipo de equipos se hacen necesarias las siguientes herramientas:
    - Groupware.
    - Software Videollamadas o chat.
    - Software de gestión de proyectos ágiles.
    - Herramientas de software que simulan la funcionalidad de Scrum boards.

Equipos multifuncionales

Los equipos multifuncionales (también llamadas células ágiles) tienen todas las competencias y habilidades necesarias para llevar a cabo el trabajo sin depender de otras personas que no formen parte del equipo.

Contrario a lo que piensa un equipo multifuncional, no se trata de que todos sus integrantes hagan de todo, se trata de que los integrantes adquieran conocimiento en distintas disciplinas (aplicables a los proyectos de la organización) y así puedan contribuir eficazmente con la colaboración.

La realidad es que incluso cuando un equipo sea experto técnico, siempre necesitarán capacitación adicional, así es que el Product Owner deberá decidir si aprobará el dinero y el tiempo para capacitarse o por el contrario serán los miembros del equipo quienes se encargarán del tema.

Colaboración con el cliente

En los proyectos tradicionales, los clientes por lo general se mantenían a distancia y solo se involucraban al principio y al final del proyecto. En Scrum es altamente recomendable que el cliente participe de las revisiones del producto y brinde retroalimentación en todos los puntos de "inspección y adaptación". Esto minimiza el riesgo y le brinda más opciones al cliente y a las partes interesadas.

Por ejemplo, en otros Marcos Ágiles como XP, es obligatorio que el Cliente forme parte del equipo.

El cliente (o sus representantes) deberían trabajar junto al Product Owner para definir las historias de usuario y detallar dichas historias antes o durante las reuniones de planificación.

El cliente y las partes interesadas por lo general participan en la Reunión de Revisión de los Sprint y, dependiendo de la relación entre el cliente y el Product Owner, el Cliente incluso podría participar de algunas reuniones de Retrospectiva de los Sprint.

Gestionar el conocimiento

Dado que los equipos autogestionados se responsabilizan de su aprendizaje y crecimiento, la gestión del conocimiento se convierte en un aspecto crucial. Esto puede tomar la forma de compartir habilidades, aprender nuevas tecnologías o técnicas, o incluso la formación entre pares.

Gestionar el conocimiento permitirá identificar, recopilar, organizar, transferir y retener el conocimiento necesario para dar soporte a todo el personal en sus actividades laborales, para la toma de decisiones bien fundadas y para aumentar la productividad.

□□ Tip para el Developer

Si bien es esencial documentar, es fundamental preguntarte: **"¿Esta documentación agrega valor?".** Evita la creación de documentos extensos y en su lugar, apunta a documentaciones claras y concisas que sean realmente útiles.

Motivación del equipo

Los equipos Scrum se caracterizan por mantener un enfoque hacia la entrega frecuente de resultados; y aunque los miembros del equipo son conscientes de la responsabilidad que esto implica, existe un factor de fondo que facilita el impulso y el esfuerzo para cumplir con los objetivos: la motivación.

La motivación hace referencia a que los miembros del equipo mantengan determinada conducta y estado de ánimo que propicien las interacciones sanas y el alto rendimiento en el proyecto.

- **Motivación Intrínseca:** este tipo de motivación es propio de cada persona, es decir que por su propia voluntad e inspiración es capaz de mantener una conducta específica y el impulso necesario para cumplir con una meta que brinda satisfacción interna y realización personal. (Ref: CHAMPFROGS – Moving Motivators)
- **Motivación Extrínseca:** este tipo de motivación hace referencia a mantener una conducta específica para responder a un impulso externo, es decir que en este caso la voluntad e inspiración de la persona se ven influenciadas por una recompensa externa (que puede ser algo físico, monetario o psicológico).

Según cómo se maneje la motivación en el equipo Scrum, eventualmente la motivación extrínseca tiende a convertirse en motivación intrínseca, pues los miembros del equipo van adaptando su conducta y mejorando su rendimiento para cumplir los objetivos sin necesidad de que todo el tiempo estén recibiendo recompensas o algo a cambio.

---

2. Simplicidad

Scrum no sería considerada una metodología ágil de no ser por su simplicidad, es por ello que se intenta al máximo reducir la burocracia en sus prácticas, se trabaja con los artefactos que son esenciales para el equipo y se sigue un flujo de prácticas simple, sin descuidar todos los elementos

necesarios para la correcta gestión del producto, eliminando cualquier trabajo superfluo o que no aporte valor.

□□ Tip para el Product Owner

Simplicidad en la Arquitectura del Producto: Una arquitectura bien diseñada y simple facilita la adición de nuevas características y la corrección de errores. La sobrecarga arquitectónica puede ralentizar considerablemente el desarrollo.

A través de la simplicidad, se puede acelerar la entrega, mejorar la comunicación y reducir los riesgos asociados con la complejidad.

Algunos elementos que hacen posible la Simplicidad en Scrum son:

Maximizar el trabajo no realizado

La simplicidad no solo trata de hacer las cosas de manera simple, sino también de identificar y eliminar cualquier tarea que no sea esencial para alcanzar los objetivos del producto. Es una forma de ser eficiente y focalizado, dedicando recursos solo a lo que realmente importa.

Clara definición de "Hecho" (terminado)

Una definición clara y concisa de lo que significa "hecho" para un elemento del producto ayuda a evitar el trabajo innecesario. Si el equipo entiende perfectamente cuándo un ítem está completo, se evita el esfuerzo extra y las revisiones innecesarias.

Priorización

La simplicidad se logra también a través de una adecuada priorización de las tareas. Al centrarse en lo que realmente aporta valor al cliente o usuario final, el equipo puede evitar la complejidad y concentrarse en lo esencial.

Comunicación efectiva

Las soluciones simples a menudo emergen de una comunicación clara y efectiva. Al fomentar un diálogo abierto y honesto entre los miembros del equipo y con los interesados, se pueden identificar y eliminar obstáculos, malentendidos y redundancias.

□□ Tip para el equipo

Cuanto más dependencias existan entre tareas o equipos, más complejo se vuelve el trabajo. Al minimizar estas dependencias, no solo se simplifica el proceso, sino que se mejora la velocidad y fluidez del trabajo.

Refactorización continua

La simplicidad también está en el código. Refactorizar regularmente el código para mantenerlo limpio y manejable es esencial. Un código más simple es más fácil de entender, modificar y mantener.

Limitar el trabajo en progreso (WIP)

Al limitar la cantidad de trabajo en progreso, los equipos pueden centrarse en finalizar tareas antes de comenzar nuevas. Esto reduce la complejidad y facilita una entrega más rápida y eficiente.

Usar herramientas de software

Las herramientas de software dedicadas a la gestión de proyectos ágiles se han consolidado como elementos cruciales para simplificar procesos y tareas. Estas herramientas ofrecen diversas ventajas, entre las cuales destacan:

- **Centralización de Datos:** Agrupan toda la información de los proyectos en un solo lugar, lo que facilita el acceso, control y análisis de la información.
- **Automatización de Tareas:** Las herramientas permiten la automatización de diversas actividades, tales como:
  - Realizar estimaciones basadas en datos históricos.
  - Calcular métricas como la velocidad del equipo.
  - Generar gráficos que permiten el seguimiento del presupuesto, el progreso del Sprint (Burndown) y el avance global del proyecto (Diagrama de flujo acumulado), entre otros.
  - Crear registros detallados de las reuniones.
- **Alertas Proactivas:** Envían notificaciones sobre elementos del Product Backlog que podrían estar retrasados o requieran atención.
- **Colaboración Remota:** Facilitan la coordinación y colaboración entre equipos que están distribuidos geográficamente, asegurando que todos estén alineados en sus objetivos y tareas.
- **Especialización en Desarrollo:** Para proyectos específicos de desarrollo de software, estas herramientas ofrecen funciones adicionales como:
  - La integración continua.
  - Ejecución de pruebas automatizadas.
  - Trazabilidad completa entre el código y las historias de usuario.

□□ Tip para el equipo

No todas las herramientas sofisticadas son las mejores. A veces, las herramientas más simples, pero funcionales, pueden ser más eficaces y fáciles de usar para el equipo.

---

3. Enfoque en el valor para los interesados

Dentro del marco Scrum, centrarse en entregar valor constante y significativo para los interesados es primordial. Este pilar se fundamenta en la premisa de que la satisfacción del cliente debe ser prioritaria y continua.

Para asegurarse de que el equipo está en la dirección correcta, es esencial tener en cuenta los siguientes puntos:

- **Entregar incrementos "Terminados":** Asegurarse de que cada incremento del producto cumpla con la Definición de Terminado (DoD) garantiza que los resultados no sólo son completos, sino que también son de alta calidad y aptos para su uso.
- **Priorizar los elementos de mayor valor:** La priorización efectiva del Product Backlog, garantiza que el equipo se enfoque en las características y funciones que ofrecen el mayor retorno de inversión y satisfacción para los interesados.
- **Validar los prototipos con los interesados:** Antes de iniciar el desarrollo completo, es crucial obtener feedback de los interesados sobre los prototipos. Esta práctica asegura que el producto esté alineado con las expectativas y necesidades del cliente desde una etapa temprana.
- **Validación continua en la Revisión del Sprint:** Presentar y validar cada incremento con los interesados durante la Revisión del Sprint permite recolectar retroalimentación directa y realizar ajustes en tiempo real, asegurando así que el producto final cumpla con sus expectativas.

□□ Tip para el Scrum Master

Reconoce y celebra cuando el equipo entregue valor real a los interesados. Esto motiva al equipo y refuerza la cultura centrada en el valor.

□□ Tip para el Product Owner

Después de la entrega de un incremento o producto completo, realice encuestas para comprender qué tan satisfechos están los interesados y qué áreas se pueden mejorar.

---

4. Disciplina

La disciplina es esencial en Scrum para garantizar que el equipo trabaje armoniosamente y cumpla con los objetivos trazados. Esta rigurosidad no sólo asegura la satisfacción del cliente sino también el funcionamiento fluido del equipo.

Lineamientos que forman parte de la disciplina del equipo:

- Lineamientos establecidos por el Equipo Scrum.
- Lineamientos establecidos por la organización (reglas de negocio).
- Bloques de tiempo asignados a los eventos Scrum (time-box).

- Definición de terminado (DoD) y criterios de aceptación.

Tip para el Scrum Master

La disciplina no significa rigidez: Si un lineamiento o práctica no está funcionando, el equipo debe sentirse empoderado para discutirlo y hacer los ajustes necesarios.

Lineamientos establecidos por el equipo Scrum

Estos son los acuerdos o reglas internas que el Equipo Scrum define para garantizar una colaboración y comunicación eficaces entre sus miembros. Por ser autoimpuestos, estos lineamientos refuerzan el pilar de auto-gestión y aseguran que el equipo se adueñe de su proceso. Gracias al pilar de la auto-gestión, son los miembros del Equipo Scrum, quienes establecen sus propios lineamientos, claro, considerando el cumplimiento de las reglas establecidas por la organización.

Los lineamientos del equipo suelen registrarse en un “Plan de Colaboración del equipo Scrum”. Estos lineamientos pueden abordar aspectos como los métodos y herramientas de comunicación preferidos, los horarios de las reuniones, la asignación de roles específicos dentro del equipo, penalizaciones por incumplimientos, los valores del equipo, o las normas sobre cómo y cuándo se realizarán ciertas tareas.

Tip para el Scrum Master

Los lineamientos pueden requerir ajustes a medida que el equipo madura. Dedique tiempo en las retrospectivas para revisar y actualizar estos acuerdos.

Lineamientos establecidos por la organización (reglas de negocio):

Estas son las directrices o reglas que la organización establece, y que todos los equipos, incluidos los Equipos Scrum, deben seguir. Pueden abordar áreas como los estándares de codificación, las políticas de seguridad de la información, los procedimientos de aprobación y revisión, entre otros. Estas reglas aseguran que, aunque el equipo tenga autonomía, su trabajo sigue estando alineado con los objetivos y requisitos más amplios de la organización.

Bloques de tiempo (time-box)

El marco de Scrum hace un uso extensivo del concepto de time-boxing, donde cada evento tiene una duración máxima asignada. Este enfoque garantiza que las ceremonias o eventos no se prolonguen innecesariamente y que el trabajo se realice de manera puntual y eficiente. Respetar estos bloques de tiempo es crucial para mantener la disciplina y el ritmo del equipo.

Algunas ventajas de establecer y cumplir los Bloques de Tiempo asignado son los siguientes:

- Se evita que el equipo pierda motivación.

- Menos gastos generales en el equipo.
- Se garantiza una alta velocidad para los equipos.
- Las prácticas relacionadas con el desarrollo de entregables son más eficientes.

> **Recuerda los bloques de tiempo de los eventos de Scrum:**
>
> - Sprint: 1-4 Semanas
> - Planificación: 8 Horas para sprints de 1 mes
> - Daily: 15 Minutos máximo
> - Revisión: 4 Horas para sprints de 1 mes
> - Retrospectiva: 3 Horas para sprints de 1 mes

Definición de terminado (DoD) y criterios de aceptación

La Definición de Terminado es una lista acordada de criterios que deben cumplirse para que una tarea o característica se considere "terminada". Asegura que todo el trabajo cumpla con un estándar mínimo de calidad y esté listo para ser entregado al cliente o puesto en producción.

Los criterios de aceptación, por otro lado, son requisitos específicos que una tarea o característica debe cumplir para ser aceptada por el cliente. Estos criterios ayudan a aclarar las expectativas y garantizar que el equipo esté trabajando hacia el mismo objetivo.

Tip para el Developer

Aunque puede ser tentador manejar múltiples tareas a la vez, la multitarea puede disminuir la calidad y eficiencia del trabajo. Enfócate en una tarea a la vez y asegúrate de que se complete según la Definición de Terminado.

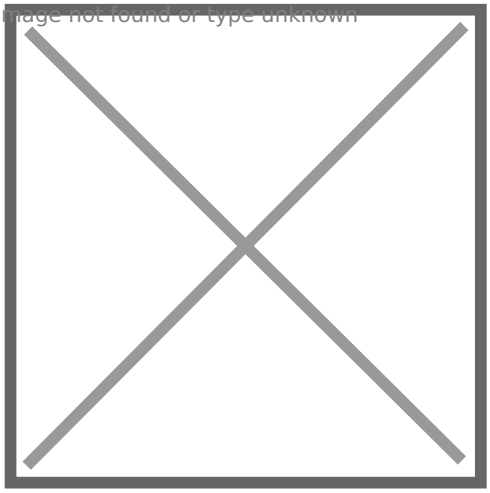
---

5. Desarrollo iterativo

Dentro de Scrum, el proceso de trabajo se lleva a cabo mediante ciclos repetitivos conocidos como Sprints. Estos Sprints son el núcleo del enfoque iterativo de Scrum, permitiendo flexibilidad y adaptabilidad a lo largo del proyecto.

A diferencia de un enfoque de desarrollo tradicional, que busca tener un plan detallado desde el comienzo, el enfoque iterativo reconoce la naturaleza cambiante del desarrollo de productos y servicios. Al adoptar un enfoque de desarrollo iterativo, Scrum permite a los equipos responder a los cambios y adaptarse a las necesidades emergentes de los clientes, el mercado y la organización.

Image not found or type unknown



Ventajas del Desarrollo Iterativo:

- Entregas Incrementales: El desarrollo iterativo da lugar a entregas de producto por etapas, ofreciendo valor a los usuarios y partes interesadas desde fases iniciales del desarrollo.
- Adaptabilidad: Dada su estructura cíclica, permite ajustes y refinamientos en cada Sprint basados en el feedback obtenido, haciendo que el producto final esté más alineado con las necesidades reales del usuario.
- Aprendizaje Continuo: Las retrospectivas al final de cada Sprint brindan al equipo la oportunidad de reflexionar sobre su desempeño y hacer mejoras continuas en su proceso.
- Estabilidad y Control: La estructura de Sprint proporciona hitos regulares para revisar y ajustar el progreso, ofreciendo puntos de control y estabilidad al equipo.
- Transparencia: Las revisiones frecuentes de Sprint aseguran que las partes interesadas estén constantemente informadas sobre el progreso y estado del producto, reduciendo la incertidumbre y construyendo confianza.

□□ Tip para el Scrum Master
Promueve una cultura donde cada miembro del equipo esté comprometido con la mejora continua, no sólo en términos del producto sino también en términos de procesos y colaboración.

3.4 - El Scrum Master y los eventos

En Scrum cada evento (ceremonia) tiene un propósito específico, diseñado para maximizar la transparencia, la colaboración y la eficiencia. **El Scrum Master,** a menudo considerado como el guardián de la agilidad del equipo, juega un papel esencial en estos eventos. Su responsabilidad no se limita a la simple organización o facilitación; va mucho más allá, actuando como un catalizador que asegura que cada ceremonia genere el máximo valor, alineando al equipo con los valores de Scrum.

En esta sección, exploraremos en detalle cómo el Scrum Master navega por cada evento, qué se espera de él y cómo su participación activa puede marcar la diferencia en la dinámica y resultados del equipo Scrum.

El Scrum Master en la Planificación del Sprint

- Apoya al Product Owner a moderar la participación de todos
- Cuida que se cumpla con la agenda y duración del evento
- Asiste al Product Owner en la definición del trabajo a realizar
- Apoya en la resolución de dudas que tengan los Developers
- Apoya a los Developers al momento de realizar las estimaciones del trabajo

> **Caso práctico:** Durante una sesión de planificación del sprint, algunos desarrolladores muestran confusión sobre una historia de usuario. El Scrum Master, Pablo, interviene: "Vamos a aclarar esta historia antes de continuar". Facilita una sesión de preguntas rápidas con el Product Owner, María, eliminando las dudas. Cuando llega el momento de las estimaciones, Pablo propone un "Planning Poker", logrando consenso. Al finalizar, recalca la importancia de mantener la comunicación abierta durante el Sprint.

□□ Tip para el Scrum Master

Asegúrate de que todas los elementos seleccionados para el Sprint estén claros y bien definidos. Si notas que hay dudas o incertidumbre, tómate un momento para clarificarlas con el Product Owner antes de que el equipo empiece a estimar.

El Scrum Master en el Daily

- Apoya a los Developers en la realización del evento, y que se respete su duración
- Identifica los impedimentos que puedan encontrar los Developers
- Asegura que el Daily se lleve a cabo a la misma hora y en el mismo lugar
- Si el Scrum Master está trabajando activamente en los elementos del Sprint, entonces participa como un Developer

> **Caso práctico:** Al tercer día del Sprint, los Developers se reúnen puntualmente para el Daily Scrum. Pablo, el Scrum Master, se asegura de que todos estén presentes y de que se inicie a tiempo. Cuando Sofía, una Developer, menciona que está teniendo problemas técnicos con una herramienta, Pablo toma nota del impedimento para tratarlo después del Daily. Aunque Pablo está trabajando en una tarea específica de este Sprint, mantiene su participación breve y centrada, como si fuera otro Developer. Al finalizar, Pablo recuerda a todos la importancia de ser consistentes con la hora y el lugar del Daily, reforzando la rutina y previsibilidad.

□□ Tip para el Scrum Master

Si durante el Daily un Developer empieza a profundizar demasiado en un impedimento, tú como Scrum Master debes intervenir amablemente y recordar al equipo que el Daily tiene una duración limitada. Sugiere abordar el impedimento en detalle justo después del Daily o programar un momento específico para tratarlo, asegurando que el foco del Daily se mantenga en la actualización rápida de cada miembro del equipo.

El Scrum Master durante el desarrollo del Sprint

- Lleva el seguimiento al progreso de los Developers con respecto a lo planificado
- Gestiona y resuelve impedimentos que puedan aparecer (en su mayoría identificados en el Daily)
- Gestiona la resolución de conflictos que puedan surgir entre los Developers
- Trabaja en estrategias que aumenten la motivación de los Developers
- Asegura que los Developers no son interrumpidos durante el Sprint
- Cuida que los Developers cumplan con los criterios de aceptación y terminado (DoD)

> **Caso práctico:** A mitad del Sprint, Pablo, el Scrum Master, nota que el equipo está un poco detrás de lo planificado. Un impedimento principal es la falta de acceso a un servidor que se identificó en el Daily. Pablo toma la iniciativa, habla con el equipo de infraestructura y resuelve el acceso al servidor al día siguiente.
>
> Días después, observa una discusión entre Sofía y Jorge (dos Developers) sobre la mejor manera de implementar una función. Pablo facilita una breve sesión para comprender los puntos de vista y logra que ambos lleguen a un acuerdo, manteniendo el ambiente armonioso.

> Finalmente, al aproximarse el final del Sprint, Pablo realiza pequeñas revisiones con los Developers para asegurarse de que todos los elementos del trabajo estén alineados con los criterios de aceptación y la Definición de Terminado (DoD).

Tip para el Scrum Master

Mantente disponible y accesible. Los impedimentos pueden surgir en cualquier momento, y tu capacidad para actuar rápidamente puede hacer la diferencia entre un Sprint exitoso y uno problemático.

El Scrum Master en la Revisión del Sprint

- Apoya al Product Owner a moderar la participación de todos
- Cuida que se cumpla con la agenda y duración del evento
- Asegura que se presentan los resultados logrados en el Sprint
- Asegura que el progreso respecto al Objetivo de Producto es discutido en esta sesión
- Recopila información que permita medir el desempeño del equipo

> **Caso práctico:** Al finalizar el Sprint, es momento de la revisión. Pablo, el Scrum Master, asegura que la sala de reuniones está lista y que todos los recursos tecnológicos funcionan correctamente para evitar interrupciones.
>
> Mientras Carla, el Product Owner, se prepara para revisar las características que completaron los Developers durante el Sprint, Pablo se mantiene alerta para moderar y garantizar que se respeten los tiempos y que todos los miembros tengan la oportunidad de participar.
>
> Pablo también se encarga de recopilar comentarios y preguntas que surgen durante la presentación, y se asegura de que, al finalizar, el equipo y los stakeholders (si los hay) discutan el progreso en relación con el Objetivo de Producto.
>
> Una vez terminada la revisión, Pablo recoge feedback sobre la revisión, buscando oportunidades de mejora para las próximas sesiones.

Tip para el Scrum Master

No te enfoques únicamente en lo que se completó: Asegúrate de celebrar los logros del equipo y reconoce los esfuerzos. Al mismo tiempo, fomenta una discusión abierta sobre lo que no se logró y las razones detrás de ello, para crear un ambiente de aprendizaje continuo.

El Scrum Master en la Retrospectiva del Sprint

- Guía y modera el evento
- Aplica estrategias y técnicas para llevar a cabo la retrospectiva
- Brinda la confianza y seguridad para que el equipo sea sincero con respecto a la reflexión sobre los resultados del Sprint
- Asiste el equipo en la identificación de lecciones aprendidas
- Guía la reflexión del equipo acerca de su rendimiento (tal como la velocidad y el progreso)

> **Caso práctico:** Tras la Revisión del Sprint, Pablo, el Scrum Master, organiza la Retrospectiva. Pablo comienza recordando la importancia de la sinceridad y el respeto mutuo durante el evento, y decide emplear la técnica "Estrella de Retro" para abordar distintas facetas del trabajo del equipo. Como facilitador, cuando un punto particularmente delicado sale a la luz, Pablo interviene para asegurarse de que la discusión sea productiva y enfocada en las soluciones, y va tomando notas de lecciones aprendidas y áreas de mejora. Al final, se asegura de que todos los acuerdos se registren y sean visibles para el próximo Sprint. Agradece al equipo por su compromiso y les anima a implementar las mejoras acordadas.

□□ Tip para el Scrum Master

Cultiva un ambiente seguro y de confianza: Recuerda que la Retrospectiva no es un espacio para señalar errores individuales, sino para identificar áreas de mejora como equipo. Fomenta la empatía y escucha activa, evitando juicios prematuros o asumir posturas defensivas.

3.5 - Técnicas útiles para el Scrum Master

Para guiar el Daily Scrum

Una de las dudas al momento de llevar a cabo el daily es saber quién se supone que debe hablar primero y en qué orden, cómo mantener la atención, por lo que se proponen las siguientes estrategias:

Pasa la ficha (token de habla)

Esta técnica busca estructurar la comunicación y asegurar una participación activa y ordenada de todos los miembros. Se utiliza un objeto (pelota, juguete, etc.) para determinar quién habla a continuación. Sólo la persona que tiene el objeto puede hablar y lo entrega aleatoriamente a otra persona para asegurar que todos prestan atención. Con esta estrategia se evitan interrupciones, se asegura que todos sean escuchados y puede agregar un elemento de diversión a la reunión que puede aliviar la tensión y hacer que la reunión sea más amena.

Toma una carta

Esta es una técnica útil para evitar la monotonía y dar a todos una oportunidad de hablar en diferentes órdenes cada día. Al comienzo del Daily, se reparten cartas numeradas a los miembros del equipo. El número en la carta determinará el orden de habla.

Caminar por el tablero

En lugar de que cada miembro del equipo simplemente responda las tres preguntas tradicionales, el equipo puede "caminar" por el tablero de Scrum, abordando cada historia de usuario o tarea, desde la más alta prioridad hacia abajo (será imprescindible que el tablero esté actualizado). Con esta estrategia se mantiene el enfoque en el trabajo real, se pueden identificar bloqueos de manera efectiva y se garantiza que las tareas más críticas sean discutidas.

Estas técnicas no son excluyentes y se pueden combinar según las necesidades del equipo. Lo más importante es mantener el propósito del Daily Scrum: asegurarse de que el equipo esté alineado, identificar bloqueos y tomar medidas para avanzar eficazmente hacia el objetivo del sprint.

Recomendaciones adicionales para el Daily Scrum

A continuación se presentan recomendaciones para mejorar la ejecución del daily, lograr que se cumpla con los objetivos de este evento, y mantener la atención del equipo:

- **Ponerse de pie:** Si el equipo está físicamente presente, considera formar un círculo o semicírculo para que todos puedan verse y escucharse claramente. Ponerse de pie conlleva a la incomodidad física lo que le recuerda a los asistentes que el daily está demorando demasiado.
- **Cronometrar el evento:** Para limitar la duración del evento y acondicionar al equipo para mantener el enfoque en el objetivo. Esto mantiene el ritmo de la sesión, garantiza la eficiencia y evita divagaciones.
- **Rotar al facilitador:** Cada día, un miembro diferente del equipo puede facilitar el Daily Scrum. Esto puede rotar en un orden definido o ser aleatorio, lo que permite eliminar la dependencia con un único facilitador o con el SM para poder iniciar el daily. Rotar al facilitador fomenta la responsabilidad compartida, introduce variedad y permite que diferentes voces guíen la reunión.
- **Evitar las largas historias:** La información concisa hace que el equipo mantenga la atención en lo que se habla en el daily.

Para guiar la Retrospectiva del Sprint

Es muy común que el equipo pueda perder el interés en participar en la retrospectiva por considerarla aburrida, por lo que existen diversas técnicas que facilitan la transparencia y la adaptación.

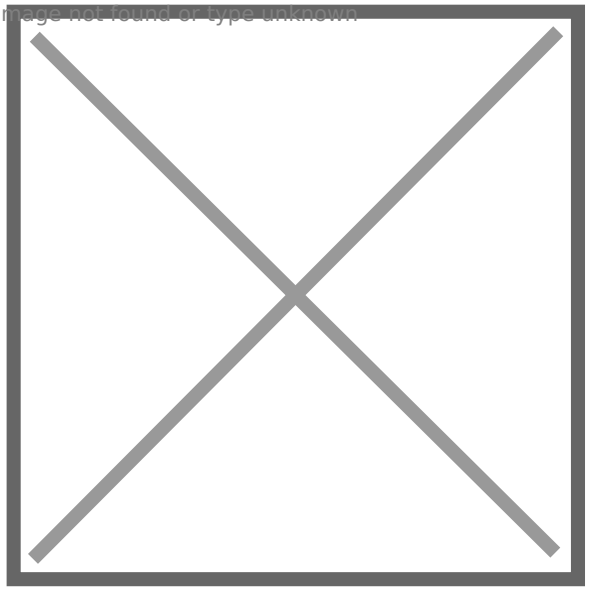
- No existe una técnica mejor que otra. Hay técnicas que funcionan muy bien en un equipo y fracasan en otro, por lo que la recomendación es que se prueben varias técnicas.
- No todas las dinámicas potencian lo mismo. El momento, la situación, e incluso la personalidad del equipo, son cosas que se tienen en cuenta a la hora de elegir la dinámica a usar.
- Es necesario que, al terminar la retrospectiva se concluya con una o varias acciones concretas y realizables

Retrospectiva de las 4L's

Algunos miembros del equipo suelen decir cosas como "si tan solo tuviéramos...", "Ojalá tuviéramos...", o tal vez digan cosas como "adivina lo que descubrí ...", "realmente me encantó ..."

La retrospectiva de 4Ls está diseñada para que las personas compartan esos pensamientos con el objetivo de mejorar continuamente.

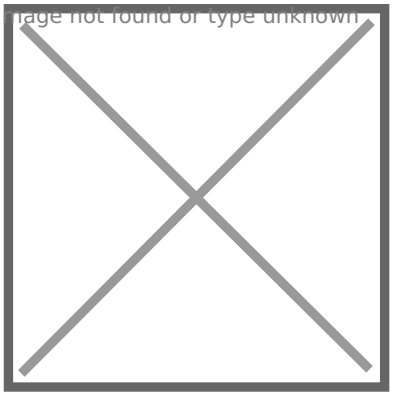
- **Nos gustó:** ¿Qué le gustó al equipo del último Sprint? Podría ser cualquier cosa, desde un proceso, un logro, una acción de equipo en particular o incluso una tecnología.
- **Aprendimos:** ¿Qué cosas aprendió el equipo de experimentos, pruebas, conversaciones y del trabajo entre ellos?
- **Nos faltó:** ¿Qué parecía faltar en el último Sprint?
- **Anhelamos:** Es algo que desearían que existiera o que fuera posible para asegurar el éxito del trabajo.



Retrospectiva del barco rápido (speed boat)

Speed Boat es una técnica en la que los miembros del equipo simulan que son tripulantes en un barco, el cual debe llegar a una isla (que representa el Objetivo del Producto). Es una forma visual y metafórica de identificar lo que ayuda y obstaculiza al equipo.

- Los remos nos ayudan a avanzar (cosas que hacemos bien)
- Los motores nos ayudan a llegar eficientemente a la isla (cosas por adoptar o mejorar)
- Los anclajes nos impiden llegar a la isla (cosas que hacemos mal)



Nota: Se pueden usar notas adhesivas para identificar cada aspecto.

Comenzar, Detener, Continuar (Start, Stop, Continue)

Los miembros del equipo escriben acciones que les gustaría que el equipo comience a hacer, detenga y continúe haciendo. Esta técnica es sencilla, pero efectiva, para identificar rápidamente áreas de mejora y aquellas en las que el equipo está yendo bien.

- **Start/Comenzar:** Aquellas tareas o prácticas que parecen ser positivas para conseguir los objetivos trazados en el equipo y que aún no se están haciendo
- **Stop/Parar:** Aquellas prácticas que no son tan eficientes o que no están dando los resultados pensados y que se deberían dejar de hacer
- **Continue/Continuar:** Todo aquello que se está haciendo, que funciona y que se debe seguir realizando.

Mad, Sad, Glad

Esta técnica busca revelar las emociones del equipo sobre eventos, decisiones y circunstancias que ocurrieron durante el Sprint. Se prepara un tablero dividido en tres secciones: **Mad (Enojado), Sad (Triste) y Glad (Feliz)**. Se puede usar un tablero físico con post-its o mediante herramientas digitales.

Cada miembro del equipo reflexiona sobre el sprint pasado y escribe en tarjetas o post-its individuales las cosas que les hicieron sentir enojados, tristes o felices (una tarjeta por cada sentimiento); y colocan sus tarjetas en la sección correspondiente del tablero. Cuando todos hayan terminado, uno por uno explica brevemente por qué escribió cada punto.

Una vez que todos han compartido sus sentimientos se discuten los patrones o temas recurrentes que surgen. ¿Hay muchos post-its sobre un tema particular en la sección "Mad"? Eso podría ser una señal de un problema que necesita ser abordado. Al profundizar en las causas subyacentes de los sentimientos, especialmente en los que generan emociones fuertes,puede revelar problemas más profundos o áreas de mejora.

Basándose en la discusión, se identifican las acciones específicas que el equipo puede tomar para abordar los problemas o circunstancias que causaron emociones negativas y para reforzar lo que causó emociones positivas.

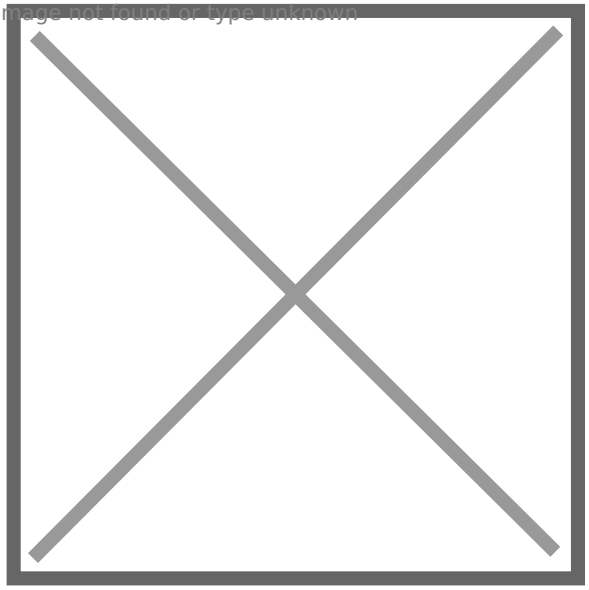
Estrella de Retro (Starfish Retrospective)

Esta técnica ayuda al equipo a reflexionar sobre diferentes áreas y decidir qué deben **comenzar a hacer, dejar de hacer, continuar haciendo, hacer más y hacer menos.**

Dibuja una estrella o estrella de mar en un tablero con cinco secciones: "Comenzar", "Detener", "Continuar", "Hacer más" y "Hacer menos". Cada miembro del equipo escribe en tarjetas o post-its sus opiniones o comentarios sobre lo que el equipo debe comenzar a hacer, dejar de hacer, etc. y los colocan en la sección correspondiente de la estrella.

Se conduce una discusión sobre cada sección. ¿Hay consenso sobre ciertos puntos? ¿Hay ideas contradictorias? Se identifican y priorizan las acciones específicas que surgieron de la discusión.

Image not found or type unknown



3.6 - Emisores de información útiles para el Scrum Master

El término "emisor de información" no hace parte de los conceptos core de Scrum. Sin embargo, en el contexto de la comunicación y la gestión ágil de producto, los emisores de información son herramientas diseñadas para mostrar el estado actual de los Sprints o del producto en su totalidad.

Estos instrumentos ofrecen una visualización clara y concisa de tareas, progresos y obstáculos, facilitando la transparencia y permitiendo un seguimiento y control efectivos. Sirven como medios esenciales para que todos los involucrados, desde el equipo Scrum hasta los stakeholders, tengan una comprensión clara del avance y estado del trabajo.

El Scrum Board

El Scrum Board o también conocido como tablero Scrum es una adaptación del tablero Kanban que nos permite dar seguimiento a cada Sprint Backlog dentro del equipo Scrum.

El Scrum Board es un emisor de información que le permite al Equipo garantizar la transparencia en los Sprints, mantiene la coordinación y permite realizar al Scrum Master sus labores de Inspección.

Comúnmente está compuesto por 4 columnas: Pendiente (to do), En Progreso (in progress), En Prueba (testing), Terminado (done).

scrum-board
Image not found or type unknown

Cada elemento por desarrollar debe ponerse en una tarjeta individual, y dado que el Scrum Board se actualiza constantemente, todas las tarjetas deberán pasar por las 4 columnas. (No se pueden hacer saltos de columnas).

Aclaraciones sobre el Scrum Board

- Cuando el tablero está físico en el lugar de trabajo del equipo, generalmente las tarjetas que se manejan son “sticky notes” de distintos colores.
- Cuando se van a desarrollar múltiples historias de usuario en un solo Sprint conviene dividir el tablero en filas, donde cada fila representa un componente o épica del producto.
- Al final de un Sprint se “borra” o reinicia el Scrum Board (para así poderlo usar para el próximo Sprint)
- Para considerar un elemento como “terminado” y moverlo a esta zona en el Scrum Board, se debe considerar la “Definición de terminado (DoD)”.
- Para facilitar la lectura del Scrum Board, a cada miembro del equipo puede asignársele un color de tarjetas (a manera de convenciones).
- Aunque por lo general el Scrum Board se actualiza durante la reunión diaria, cada Developer tiene autonomía para actualizar su conjunto de elementos asignados en el Sprint Backlog.

Burndown chart del Sprint

El “Burndown Chart” es un emisor de información que muestra la cantidad de trabajo pendiente en el Sprint actual en función del tiempo.

Es un emisor gráfico de 2 dimensiones:

- El eje vertical muestra el trabajo restante, que se determina con la sumatoria de puntos de historia a desarrollar en el Sprint
- El eje horizontal muestra el tiempo, es decir la duración del Sprint en días hábiles.

burndown-chart
Image not found or type unknown

Al inicio del Sprint, la cantidad de trabajo restante es la más alta y, a medida que se avanza el Sprint, esta cantidad debería disminuir, "quemándose" (burn) el trabajo hasta llegar a cero al final del Sprint.

El Burndown Chart es una herramienta esencial para los equipos Scrum ya que les permite visualizar diariamente su progreso, identificar posibles problemas y ajustar su ritmo de trabajo si es necesario para cumplir con el Objetivo del Sprint (Sprint Goal).

Aclaraciones sobre el Burndown Chart

- Una posible variación de esta herramienta es el **"Burnup chart" que muestra gráficamente la cantidad de trabajo completado** a lo largo del tiempo, en lugar de lo que queda por hacer (a medida que avanza el tiempo, la línea del gráfico asciende, mostrando cuánto trabajo se ha "acumulado" o completado).
- Los **Developers son los responsables de la actualización del Burndown Chart.** El Scrum Master puede facilitar y asegurarse de que el Burndown Chart se mantenga actualizado, pero no es su responsabilidad directa hacerlo.

- Por lo general la actualización se realiza durante los Scrum Diarios.

Diagrama de flujo acumulado

El Diagrama de Flujo Acumulado (CFD - Cumulative Flow Diagram) es un emisor de información bastante útil para la elaboración de informes y el seguimiento de los resultados del proyecto; y se originó a partir de la metodología Kanban.

Este emisor de información muestra el progreso del producto respecto a los ítems del Product Backlog, y se construye un único diagrama de flujo acumulado por producto.

Este gráfico se compone de dos ejes; el eje vertical "Y" muestra la cantidad de ítems (como tareas, características, user Historias de Usuario). El eje horizontal "X" muestra el tiempo, usualmente días, semanas o meses.

Cada área coloreada en el gráfico representa un estado o categoría diferente del flujo de trabajo:

- **La zona "gris"** muestra el comportamiento del Product Backlog; y el aumento de esta zona significa la adición de nuevos requerimientos/tareas/historias de usuario. La disminución de esta zona significa que se retiran requerimientos/tareas/historias de usuario que ya no generan valor (como resultado de los cambios al producto).
- **La zona "azul"** muestra el trabajo que fue seleccionado por el equipo para desarrollar en los distintos Sprints, es decir el trabajo que se seleccionó como Sprint Backlogs.
- **La zona "verde"** muestra cuántos ítems han sido completados en el tiempo. Idealmente, deberías ver un crecimiento estable en esta zona si el trabajo se está completando a un ritmo constante. Cuando la zona azul está por encima de la zona verde, normalmente significa que el equipo seleccionó más trabajo del que podía terminar, esto se conoce como "deuda técnica".
- La diferencia entre la zona "verde" y la zona "gris", está relacionada con el progreso del producto.
- Los puntos marcados en el gráfico muestran los distintos Sprints.

Aclaraciones sobre el Diagrama de Flujo Acumulado:

- Si una zona del gráfico se ensancha mientras las demás no, es una señal de que los ítems se están acumulando en ese estado y puede indicar que hay un cuello de botella.
- Si las zonas fluyen de manera paralela y sin muchas variaciones en su ancho, indica que el proceso es estable. Si se expanden y contraen drásticamente, puede indicar que hay inconsistencias en el flujo de trabajo.
- El responsable de este emisor de información es el Scrum Master, aunque normalmente es el Product Owner quien lo utiliza para mostrar el progreso a las partes interesadas.

Registro de obstáculos/impedimentos

El registro de obstáculos, o también llamado registro de impedimentos es una herramienta que se utiliza para llevar un registro y seguimiento de los obstáculos o bloqueos que impiden o ralentizan el progreso del equipo Scrum. Estos impedimentos pueden ser cualquier cosa que obstruya el flujo de trabajo del equipo, desde problemas técnicos hasta cuestiones organizativas.

> Un impedimento es cualquier factor que impide que el equipo alcance su productividad potencial. Puede ser algo interno, como un desafío técnico o falta de habilidades específicas, o externo, como la espera de una aprobación de otro equipo o departamento.

Mantener un registro de impedimentos ayuda a asegurar que los problemas no se pasen por alto y que se les dé la debida prioridad. También proporciona una visión histórica que puede ser útil para futuras retrospectivas y para identificar patrones o problemas recurrentes.

Aclaraciones sobre el registro de impedimentos

- El Scrum Master es el rol encargado de abordar y eliminar los impedimentos que estén afectando al equipo. Sin embargo, es responsabilidad de todo el equipo identificarlos y comunicarlos.
- Durante el Scrum Diario (o Daily), los miembros del equipo informan sobre cualquier nuevo impedimento que haya surgido. El Scrum Master toma nota de estos y los añade al registro de impedimentos.
- Cada entrada en el registro de impedimentos debe tener una descripción clara del problema, la fecha en que fue identificado, su impacto potencial en el equipo o en el producto y cualquier otro detalle relevante.
- También es útil incluir el estado del impedimento (por ejemplo: identificado, en progreso, resuelto) y las acciones tomadas o necesarias para resolverlo.
- En algunas organizaciones este registro es global para todos los proyectos o equipos y sirve como base de conocimiento para todos los integrantes de los distintos equipos Scrum.

Registro de lecciones aprendidas

Registrar las lecciones aprendidas es una parte fundamental para la mejora continua de las prácticas Scrum dentro de una organización. En esta herramienta se registran las prácticas,

experiencias, conocimientos y recomendaciones que favorecieron el desempeño del equipo en términos de costo, tiempo y calidad, también se registran aquellos errores que se cometieron y a los que se les dió solución. Estas soluciones hacen parte de las lecciones aprendidas, útiles para resolver problemas similares en el futuro.

Las lecciones aprendidas, además, favorecen el aprendizaje organizacional y la gestión del conocimiento. Así que las lecciones aprendidas deben almacenarse, alimentarse y transferirse de manera apropiada. Para esto, es importante que cada lección tenga una estructura para facilitar su búsqueda, como por ejemplo:

- Una identificación (ID)
- Descripción: Breve resumen de la situación o problema.
- Situación/Contexto: En qué fase o circunstancia del proyecto se encontró la lección.
- Lección Aprendida: Detalle de lo que se aprendió, ya sea un error que se debe evitar o una buena práctica que se debe repetir.
- Recomendaciones: Pasos o acciones sugeridos para futuros proyectos a partir de la lección aprendida.
- Responsables: Quiénes estuvieron involucrados o quiénes deberían ser informados sobre esta lección.
- Fecha: Cuándo se identificó la lección.
- Categoría: Por ejemplo, la fase del ciclo de vida del proyecto

Aclaraciones sobre el registro de lecciones aprendidas:

- Este artefacto normalmente es una de las responsabilidades del Scrum Master.
- Los equipos suelen tener reuniones específicas para discutir y documentar lecciones aprendidas; que pueden darse durante las sesiones de retrospectiva o en revisiones post-proyecto (cuando se manejan enfoques más tradicionales).
- Es crucial que estas sesiones se lleven a cabo en un ambiente abierto y sin culpas, para fomentar la sinceridad y el aprendizaje genuino.
- Al iniciar un nuevo proyecto o la construcción de un producto, el equipo debe revisar registros anteriores para identificar posibles riesgos, desafíos o mejores prácticas relevantes.
- La organización o el equipo debe tener un sistema para almacenar y acceder a estas lecciones, ya sea una base de datos, un repositorio centralizado o incluso herramientas digitales especializadas.

Portafolio de proyectos

El portafolio de proyectos consiste en un registro de información histórica, esta información podría ser útil (por ejemplo) cuando la organización empieza un proyecto similar a uno que haya ejecutado en el pasado y por tanto algunos elementos sean similares, esto reduce el tiempo de planificación, desarrollo, revisión, implementación y/o cierre, además de hacer uso de las lecciones aprendidas y brindar los insumos para la gestión del conocimiento.

Al igual que las lecciones aprendidas, el portafolio de proyectos debería tener información detallada y organizada que permita su alimentación, almacenamiento y transferencia. Un software específico para la gestión del portafolio de proyectos es útil, por lo que una organización debería considerar esta posibilidad.

Algunos de los elementos que podrían incluirse en el portafolio de proyectos son:

- Una identificación (ID)
- Nombre del Proyecto
- Categoría (Ej: Software – Implementación – Construcción – Investigación, Etc)
- Descripción corta del proyecto (buscando resumir la visión del mismo)
- Tiempo total de proyecto
- Costo total del proyecto
- Enlace a repositorio o herramienta de información (donde se podrá encontrar más información del proyecto)

4. Product Owner: Gestión ágil de productos

La responsabilidad del Product Owner es crucial en Scrum, fungiendo como el principal enlace entre las partes interesadas y el equipo Scrum, con la meta de maximizar el valor generado a través del desarrollo del producto. Un Product Owner posee la autoridad y el conocimiento profundo necesarios para articular y defender las necesidades y expectativas de los stakeholders con precisión. En este capítulo, exploraremos las responsabilidades, habilidades y actividades de un Product Owner. Esto brindará una comprensión ampliada que complementa lo ya discutido en el modelo Core de Scrum, orientándote en cómo un Product Owner puede liderar efectivamente la visión del producto a través de colaboración efectiva, liderazgo estratégico y toma de decisiones basada en datos. Puntos clave de este capitulo: - Comprender la importancia estratégica de la responsabilidad del Product Owner en el desarrollo de productos ágiles. - Identificar las habilidades cruciales que un Product Owner necesita cultivar para liderar con eficacia. - Profundizar en las actividades específicas y diarias que definen la labor del Product Owner. - Analizar cómo un Product Owner puede funcionar como un puente eficaz entre el equipo Scrum y las partes interesadas, garantizando una entrega de valor maximizada.

4.1 - Gestión ágil de producto

La gestión de productos, o "Product Management" en inglés, es una disciplina que se centra en guiar el desarrollo, la estrategia y la entrega de productos.

La gestión de producto se pueden manejar con enfoque tradicional o con enfoque ágil: El Agile Product Management (Gestión ágil de producto) está diseñado para entornos y mercados que cambian rápidamente, donde la adaptabilidad y la respuesta rápida son esenciales. Mientras que el Product Management tradicional puede ser más adecuado para industrias o productos donde los requisitos son menos volátiles y el desarrollo puede planificarse con bastante antelación.

> Muchas empresas modernas están adoptando un enfoque ágil, al reconocer los beneficios de la adaptabilidad y la iteración rápida.

La gestión ágil de producto permite establecer estrategias para crear productos en un entorno ágil, manteniendo un enfoque adaptable para que la organización y el equipo propiamente puedan responder rápidamente a la retroalimentación de los clientes y crear mejores productos.

A diferencia de las técnicas de desarrollo tradicionales, la gestión ágil se centra en la adaptabilidad, permitiendo que el equipo responda rápidamente a la retroalimentación de los clientes.

Tip para el Product Owner
Valora más una entrega temprana y frecuente que esperar hasta que el producto esté "perfecto".

Beneficios de la gestión ágil de producto

La gestión ágil proporciona un enfoque más flexible en comparación con la planificación y el desarrollo tradicional, lo que permite que se construyan productos en incrementos cortos.

Algunos beneficios clave de la gestión ágil de productos:

- Ajustar continuamente la hoja de ruta del producto (abordado más adelante).
- Responder rápidamente a los requisitos nuevos y cambiantes, y entregar valor de manera incremental.

- Colaborar para entregar el trabajo rápidamente y con calidad.
- Aprender de los clientes y mejorar el producto.

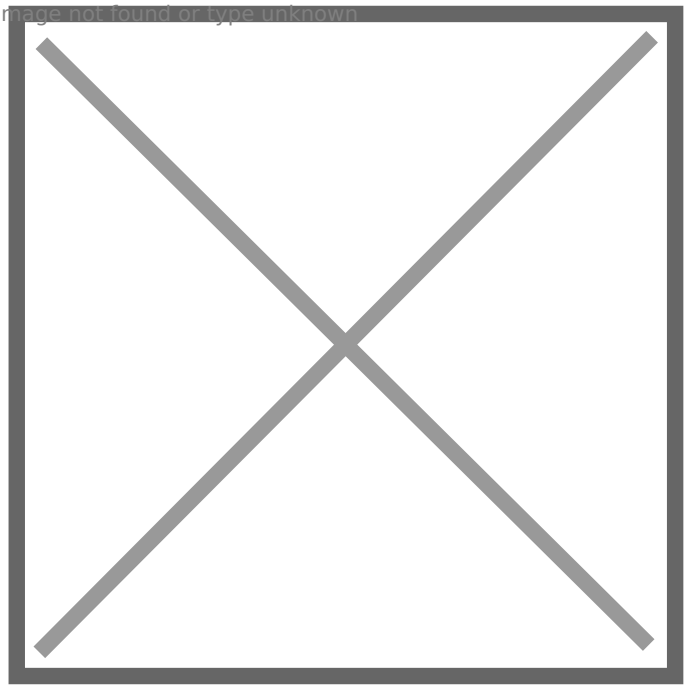
Tip para el Scrum Master

Anima al equipo a adoptar una mentalidad de mejora y aprendizaje constante, utilizando cada sprint y retroalimentación como una oportunidad de crecimiento.

Ecosistema de la gestión ágil de producto

Cada producto requiere un enfoque único y personalizado, tiene sus propios objetivos y retos, lo que se traduce en una gestión de productos basada en lograr la conexión entre las **expectativas de negocio**, una gran **experiencia de usuario** y el aprovechamiento de la **tecnología.**

Image not found or type unknown



- **UX (Experiencia de Usuario):**
  - Investigación centrada en el usuario a través de entrevistas, encuestas y pruebas de usabilidad, se recolecta feedback directo de los usuarios, lo que permite iterar y mejorar el producto continuamente.
  - Los prototipos y entregables se desarrollan, prueban y refinan de forma cíclica para garantizar una experiencia óptima para el usuario.
  - Los diseñadores UX trabajan en estrecha colaboración con los equipos de desarrollo para asegurar que la visión del producto y las necesidades del usuario se integren adecuadamente.
- **Expectativas de Negocio:**
  - En el Agile Product Management, los objetivos de negocio se definen con claridad y se comunican al equipo. Estos objetivos guían las decisiones y prioridades del equipo.
  - Se establecen indicadores clave de rendimiento para medir el éxito del producto en términos de negocio.

    - A diferencia de las estrategias rígidas tradicionales, en un entorno ágil, las estrategias y objetivos pueden adaptarse en función de la evolución del mercado y del feedback del cliente.
- **Tecnología:**
    - Se aplican técnicas como Integración Continua y Entrega Continua (CI/CD) para asegurar que las características y mejoras del producto se desarrollen, prueben y desplieguen rápidamente y de manera eficiente.
    - Se usan herramientas colaborativas (como JIRA o Trello) que permiten gestionar y priorizar las tareas y seguir el progreso en tiempo real.
    - La tecnología implementada debe ser adaptable y escalable, permitiendo que el producto crezca y se modifique según las necesidades cambiantes.

> El diseño y la investigación de la **Experiencia de Usuario** aseguran que el producto esté alineado con las necesidades y expectativas del usuario y del negocio. Las **Expectativas de Negocio** definen la dirección y los objetivos, proporcionando un marco dentro del cual el diseño y la tecnología deben operar. La **Tecnología** es el medio que permite que las expectativas respecto a la Experiencia de Usuario y del negocio se materialicen, proporcionando soluciones que cumplan con las expectativas del usuario y los objetivos de negocio.

Tip para el equipo
En escenarios de **desarrollo de software** es importante implementar técnicas de Integración Continua y Entrega Continua **(CI/CD)** para acelerar el proceso de desarrollo y despliegue.

La gestión ágil de producto en una organización

El enfoque jerárquico en comparación con el enfoque de la gestión ágil de producto abarca diferencias sustanciales en cómo se organizan, operan y toman decisiones las organizaciones:

	Enfoque Jerárquico	**Enfoque de gestión ágil de producto**
Toma de decisiones	Las decisiones vienen desde los niveles superiores de la organización y se transmiten hacia abajo. Demora en la toma de decisiones por necesidad de aprobación en múltiples niveles.	Decisiones descentralizadas tomadas por Product Owner en colaboración con su equipo y stakeholders. Se promueve la autonomía y la toma de decisiones basadas en datos.
Estructura	Departamentos o divisiones con roles claramente definidos.	Equipos multifuncionales centrados en productos o características específicas.
Comunicación	Comunicación vertical (de arriba hacia abajo y viceversa). Silos de información y desafíos en la comunicación.	Comunicación abierta y transparente entre todos los niveles y departamentos.

	Enfoque Jerárquico	Enfoque de gestión ágil de producto
Respuesta al cambio	Los cambios suelen ser más lentos y requieren aprobación en varios niveles.	Los cambios pueden implementarse rápidamente en función de la retroalimentación del usuario y las métricas del producto.
Cultura	Centrada en procesos, estructuras y cumplimiento de tareas asignadas.	Centrada en el valor para el cliente y negocio, y la entrega continua de características útiles.

Principios de la gestión ágil de producto

- **Valor sobre Funciones:** En lugar de centrarse simplemente en la cantidad de características, se enfoca en entregar el máximo valor al usuario.
- **Iteración sobre Perfección:** Liberar versiones tempranas y obtener feedback es más valioso que esperar a tener el producto "perfecto".
- **Colaboración sobre Silos:** Equipos multidisciplinarios trabajando juntos producen mejores resultados que departamentos trabajando en aislamiento.

Riesgos de no adoptar la gestión ágil de producto

- **Desarrollo Lento:** Sin un enfoque ágil, los equipos pueden quedar atrapados en largos ciclos de desarrollo sin entregas reales.
- **Desalineación con el Mercado:** Sin feedback constante, el producto puede desviarse de las necesidades reales del mercado.
- **Gastos Ineficientes:** Los recursos pueden desperdiciarse en características que no son valiosas para los usuarios.

¿Qué debo evitar en el Agile Product Management (Gestión ágil de producto)?

El Product Management es un área crucial que determina en gran medida el éxito de un producto. Sin embargo, hay ciertas trampas en las que como Product Owner puedes caer y que se deben evitar:

- **Convertirse en el 'Camarero de las funcionalidades':** Es un error común que los Product Owners se transformen en meros **intermediarios,** recogiendo y sirviendo todas las peticiones de características sin cuestionarlas. Esto puede conducir a un **ciclo interminable de desarrollo** que no resuelve los problemas reales ni cumple con los objetivos estratégicos.
- **Evitar la retroalimentación:** Ignorar el feedback de los usuarios es un **grave error.** No es obligatorio actuar sobre cada comentario, pero sí es vital escuchar, analizar y considerar la retroalimentación para mejorar el producto continuamente.
- **Visión y estrategia:** Mantén clara y presente tu visión y estrategia. **NO permitas que tus clientes decidan las soluciones**; ellos identifican problemas, pero tú (como Product

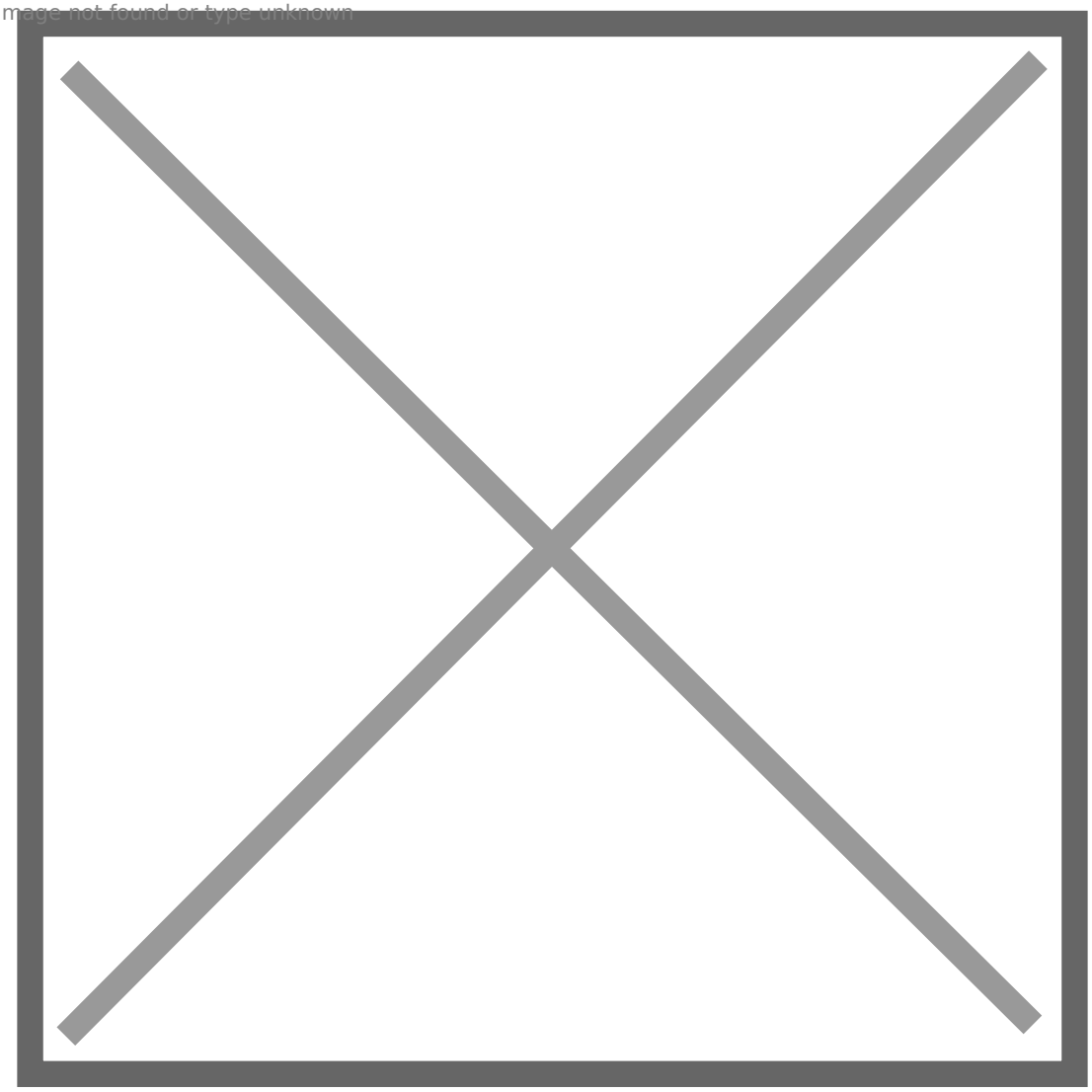
Owner) debes diseñar las soluciones.

- **Descuidar la comunicación:** Un Product Owner debe ser el puente entre diferentes equipos. La falta de comunicación clara y abierta puede llevar a malentendidos y retrasos.
- **No Establecer métricas claras:** Si no sabes qué estás midiendo, no sabrás si estás teniendo éxito. Establece KPIs claros desde el principio y monitorea el avance del equipo con respecto a lo que se espera del producto.
- **Experimenta antes de implementar:** Experimenta y valida tus ideas antes de implementarlas. Sé selectivo y prioriza las funcionalidades que realmente aportan valor.
- **No aprender del fracaso:** No todas las características o productos serán un éxito. En lugar de verlo como un fracaso, utilízalo como una oportunidad de aprendizaje.
- **Tratar de ser todo para todos: NO** puedes satisfacer todas las necesidades de todos los usuarios. Define el público objetivo y enfoca tus esfuerzos en resolver sus problemas y satisfacer sus necesidades.
- **No respetar los plazos:** Aunque la flexibilidad es una ventaja en la gestión ágil de producto, es esencial establecer y respetar plazos realistas.
- **Ignorar el mercado:** Mantente informado sobre las tendencias del mercado y la posición de tus competidores. Un producto, por más bueno que sea, si no cumple con una necesidad del mercado o no se diferencia de la competencia, puede no tener éxito.

Actores involucrados en la gestión ágil de producto

En la gestión ágil de producto existen actores complementarios al equipo Scrum, que son necesarios y usualmente definidos en las organizaciones para cubrir todas las necesidades y requerimientos de las partes interesadas:

Image not found or type unknown



Equipo Scrum:

En su concepción más básica, Scrum se centra en cómo un equipo puede trabajar de manera eficiente para entregar productos de alta calidad en ciclos iterativos y rápidos. Para hacerlo, Scrum establece tres responsabilidades: Product Owner, Developers y Scrum Master. Cada uno contribuye al flujo ágil de trabajo.

Sin embargo, cuando se trabaja con Scrum en un entorno empresarial real, con múltiples productos, interdependencias y expectativas de diversas partes interesadas, rápidamente nos damos cuenta que existen muchas otras consideraciones que no son cubiertas por el equipo Scrum.

Actores organizacionales o del cliente:

El equipo Scrum no trabaja totalmente aislado: existen roles adicionales, a menudo anclados en la estructura organizacional o en las necesidades del cliente, que desempeñan funciones críticas para asegurar que los esfuerzos del equipo Scrum se mantengan alineados con los objetivos del negocio y las expectativas del cliente.

Estos roles organizacionales o del cliente actúan como **puentes,** facilitando la comunicación, alineación y toma de decisiones entre el equipo Scrum y el ecosistema empresarial más amplio en el que operan. Entre los roles adicionales comúnmente podemos encontrar:

- *Business Manager (Gestor de negocio):* Se enfoca en la viabilidad comercial del producto, asegurándose de que el producto no solo sea funcional sino también rentable y alineado con los objetivos estratégicos de la organización.
- *Line Manager (Gestor de líneas de negocio):* Supervisa segmentos específicos o líneas de negocio y en conjunto con el Product Owner garantiza que el producto cumpla con las necesidades y objetivos de esa línea en particular.
- *Program Manager (Gestor de programas):* Coordina múltiples proyectos o equipos ágiles para garantizar que trabajen hacia objetivos comunes y que se gestionen las dependencias entre ellos.
- *APMO (Oficina de proyectos ágiles):* Se encarga de establecer, promover y asegurar las mejores prácticas ágiles a lo largo de la organización. Funciona como un centro de excelencia para las metodologías ágiles.
- *UX/UI Designers:* Aunque no siempre se mencionan en la estructura de un equipo Scrum, desempeñan un papel crucial en la gestión ágil del producto. Se encargan de diseñar una interfaz que sea agradable y funcional para el usuario.
- *Analistas de negocio:* Trabajan en estrecha colaboración con el Product Owner y el equipo Scrum para definir requisitos, entender las necesidades del negocio y traducirlas en características del producto.

□□ Tip para el Product Owner

Asegúrate de mantener una comunicación fluida con los actores externos al equipo (Business Managers, Line Managers, Analistas de Negocio) para garantizar que la visión del producto esté alineada con los objetivos de negocio.

4.2 - El Product Owner y los eventos

En Scrum, cada evento (ceremonia) está diseñado con un propósito específico para maximizar la transparencia, la colaboración y la eficiencia. **El Product Owner,** es fundamental para representar la voz del cliente y los intereses del negocio, y desempeña un papel crucial en estos eventos.

Su responsabilidad va más allá de simplemente definir y priorizar el trabajo; es el vínculo esencial que asegura que el equipo esté trabajando en las características más valiosas, alineando la entrega de producto con los objetivos del negocio.

En esta sección, exploraremos en detalle cómo el Product Owner se involucra en cada evento, qué se espera de él y cómo su aporte activo puede influir positivamente en la dinámica y los resultados del equipo Scrum.

El Product Owner en la Planificación del Sprint

- Facilita el evento, asegurando que durante la sesión se discuten 3 temas: ¿por qué este Sprint es valioso?, ¿qué se puede terminar (done) en este Sprint?, ¿cómo se realizará el trabajo elegido?
- Brinda información clara y precisa sobre el trabajo que se pueda terminar este Sprint
- Aclara las dudas en torno a las expectativas de las partes interesadas en el trabajo de este Sprint
- Aclara la información y detalles sobre elementos del Product Backlog.
- Define el Sprint Backlog y el Sprint Goal según la sesión con el equipo durante este evento
- Negocia el alcance del Sprint con los Developers.

> **Caso práctico:** Durante la planificación del sprint, el equipo está indeciso sobre qué funcionalidades priorizar. El Product Owner, Carlos, comparte recientes feedbacks de clientes y argumenta: "Nuestro principal competidor acaba de lanzar una funcionalidad similar, y debemos responder". Desglosa los elementos del Product Backlog, aclarando cualquier duda. Luego, al definir el Sprint Backlog, surge un debate sobre la carga de trabajo. Carlos, con base en la prioridad y la visión del producto, guía al equipo para acordar un Sprint Goal centrado en el impacto y valor para el usuario.

Tip para el Product Owner

La preparación es clave: Asegúrate de entender muy bien los elementos del Product Backlog que has pre-seleccionado para discutir en la Planificación del Sprint, y ten a mano toda la información relevante para aclarar cualquier duda que puedan manifestar los Developers.
También asegúrate de que las historias de usuario y las tareas estén bien definidas, claras y priorizadas. Las más prioritarias deben tener suficiente detalle para que el equipo pueda trabajar en ellas.

El Product Owner en el Daily

Nota aclaratoria: Si el Product Owner está trabajando activamente en los elementos del Sprint, entonces participa como un Developer, de lo contrario no participa en este evento.

En caso que el Product Owner participe del Daily, es posible que pueda:

- Brindar aclaraciones sobre algún elemento del Product Backlog.
- Recopilar feedback para posibles reajustes del Product Backlog.

> **Caso práctico:** Durante un Daily, el equipo de desarrolladores discute un desafío relacionado con una historia de usuario. No están seguros de cómo implementar una funcionalidad específica. Laura, la Product Owner, quien ocasionalmente asiste a los Dailies, rápidamente comprende la confusión. Después del Daily, aparta a los miembros del equipo involucrados y proporciona una aclaración detallada con ejemplos del comportamiento esperado, resolviendo el desafío. Además, toma nota para mejorar la descripción de historias similares en el futuro.

Tip para el Product Owner

Si decides asistir al daily, escucha activamente pero evita interrumpir o desviar la conversación a menos que sea estrictamente necesario. Si escuchas sobre desafíos o problemas relacionados con el Product Backlog, toma notas para refinar y mejorar el Backlog después del daily.
Si ves que hay confusiones o malentendidos sobre el Product Backlog, busca un espacio después del Daily para resolver las dudas.

El Product Owner durante el desarrollo del Sprint

- Agrega elementos al Product Backlog de acuerdo a las prioridades y expectativas de las partes interesadas.
- Realiza el refinamiento y priorización de los elementos del Product Backlog a partir de las sesiones de trabajo con el cliente y/o partes interesadas (esto suele hacerlo en paralelo mientras los Developers estan construyendo el producto).
- Resuelve las dudas a los Developers sobre el trabajo que esta en desarrollo (WIP).
- Colabora con las partes interesadas, identificando necesidades y ajustando expectativas.

>

> **Caso práctico:** Mientras el equipo está a mitad de camino en el Sprint actual, Tomás, el Product Owner, ya está en conversaciones con un stakeholder sobre las características de un futuro elemento que se desarrollará para el siguiente sprint. Tras obtener un entendimiento claro, programa una reunión de refinamiento con dos Developers para abordar este elemento. Laura, una de las Developers, sugiere una implementación que podría simplificar el trabajo. Tomás ajusta el elemento en el Product Backlog. Cuando llega el momento de la planificación del siguiente sprint, este elemento se discute con mayor fluidez y claridad, gracias a la planificación en paralelo que Tomás llevó a cabo.

□□ Tip para el Product Owner

Inicia el refinamiento y priorización de los elementos de los próximos sprints con anticipación, esto garantiza que estén listos cuando llegue el momento de la Planificación. Al trabajar en paralelo, es crucial que tanto el equipo como las partes interesadas tengan una visión clara del producto y cómo encajan las piezas.

El Product Owner en la Revisión del Sprint

- Facilita y dirige la sesión, garantizando un flujo estructurado.
- Asegura que el progreso respecto al Objetivo de Producto sea el foco de discusión de esta sesión.
- Invita y asegura que las partes interesadas clave estén presentes y activamente involucradas.
- Discute el Product Backlog tal y como está.
- Proyecta el objetivo y las fechas de entrega probables basándose en el progreso hasta la fecha (si es necesario).
- Verifica que el incremento cumpla con lo planificado, considerando tanto los criterios de aceptación como la Definición de Terminado (DoD).
- Facilita la discusión sobre posibles mejoras o ajustes en el producto.
- Confirma o redefine prioridades del Product Backlog basándose en la demostración realizada por los Developers, la deuda técnica (si existe) y en las necesidades cambiantes del negocio.
- Aborda la deuda técnica (si existe) y discute su impacto y posibles estrategias para gestionarla o reducirla.
- Indaga sobre las causas de la deuda técnica (para identificar qué aspectos deben mejorarse o ajustarse para la siguiente Planificación del Sprint).

> **Caso práctico:** Durante la revisión del sprint se presenta una funcionalidad de una aplicación de e-commerce. Un stakeholder clave menciona que, aunque la funcionalidad es lo que se pidió, el diseño podría ser más intuitivo para usuarios no técnicos. Laura, la Product Owner facilita el diálogo, para entender mejor la situación y añade esta observación al Product Backlog. Adicionalmente se

> encuentra un elemento en deuda técnica, por lo que Laura, reajusta el Product Backlog según el feedback recibido.

Tip para el Product Owner

Si se identifica deuda técnica, trata de comprender su origen y considera estrategias para gestionarla en futuros sprints. También asegúrate de registrar las observaciones y feedback para que puedan ser considerados en futuras decisiones y planificaciones.

El Product Owner en la Retrospectiva del Sprint

- Participa en la inspección de cómo fue el último Sprint con respecto a los individuos, las interacciones, los procesos, las herramientas, y la Definición de terminado (DoD).
- Participa de la identificación de lecciones aprendidas, compartiendo perspectivas desde el lado del negocio y el feedback del cliente.
- Participa en identificar las suposiciones erróneas que pudieron desviar al equipo y busca entender la causa raíz, los problemas que encontró el equipo, y cómo esos problemas fueron (o no fueron) resueltos.
- Reconoce y destaca los logros y las áreas de mejora observadas durante el Sprint.
- Puede sugerir la ampliación de la Definición de terminado (DoD) para mejorar la calidad del producto.

> **Caso práctico:** En la retrospectiva, Miguel, el Product Owner, resalta cómo el equipo logró superar un desafío técnico particular durante el Sprint. Sin embargo, surge que algunos Developers sintieron que no se comunicaron suficientemente bien los objetivos de negocio detrás de ciertas tareas. Patricia, una Developer, menciona que esto llevó a algunas decisiones técnicas que, de haber tenido toda la información, habrían sido diferentes. Miguel reconoce esta brecha y propone sesiones breves de alineación antes de cada Sprint para garantizar que el contexto del negocio esté claro. El equipo acuerda intentarlo en el próximo Sprint y evaluar su efectividad en la siguiente retrospectiva.

Tip para el equipo

La Retrospectiva **NO** es un espacio para culpar, sino para identificar oportunidades de mejora.

4.3 - Las Historias de Usuario

Las Historias de Usuario (User Story) son la forma recomendada en entornos ágiles para escribir los requisitos del producto desde la perspectiva del usuario. Dado que son afirmaciones breves, simples y fáciles de entender, resultan en una mejor comunicación entre los Stakeholders y el equipo Scrum.

Una de las principales ventajas de las Historias de Usuario es su estructura simplificada y orientada a la acción. Debido a que se centran en entender "¿quién solicita la funcionalidad?", "¿qué necesita/requiere?" y "¿por qué o para qué lo necesita?", las Historias de Usuario proporcionan una vista clara y contextualizada de lo que se espera, eliminando ambigüedades comunes en los requisitos tradicionales.

Los criterios de aceptación, que acompañan a cada Historia de Usuario, definen las condiciones específicas que deben cumplirse para que la historia se considere satisfactoriamente implementada o desarrollada. Actúan como una guía para el equipo, aterrizando las expectativas del Product Owner sobre el comportamiento o funcionalidad que debe tener esa historia. Mientras que, la Definición de Terminado (DoD) es la que establece cuándo una historia o tarea se considera completa en su totalidad.

Las Historias de Usuarios centran la atención en los usuarios finales, por lo que no utilizan un lenguaje técnico, con el fin de facilitar el entendimiento para los usuarios y asimismo ofrecer un contexto suficiente para guiar el esfuerzo de los Developers.

> Mike Cohn, uno de los pioneros en el desarrollo ágil, menciona que "Las historias de usuario permiten a los equipos trabajar en un lenguaje compartido, y eso hace que toda la experiencia de desarrollo sea más orientada al usuario y al valor. En lugar de ser una mera lista de tareas, las historias son llamadas a la acción para resolver desafíos específicos para los usuarios".

Estructura de una Historia de Usuario

Las Historias de Usuario suelen expresarse como una frase simple y breve que cumple con la siguiente estructura:

Como: Describe el ***Rol*** del Stakeholder que solicita (o usaría) la funcionalidad o requerimiento.

Quiero: Describe la ***necesidad o requerimiento*** del usuario, por lo general, es una frase corta.

Para: Describe el ***beneficio esperado*** por el Stakeholder una vez se desarrolle el requerimiento

Ejemplos de Historias de Usuario

> **Como:** Cliente, **Quiero:** pagar mi suscripción mensual vía sitio web por medio de transferencia bancaria o tarjeta de crédito, **Para:** evitar el desplazamiento hasta la sucursal de pago.

> **Como:** Supervisor de ventas, **Quiero:** consultar un listado de los pedidos de venta que han sido registrados y aún no han sido procesados, **Para:** Tener un reporte detallado a la mano.

> **Como:** Ejecutivo de cuenta, **Quiero:** consultar los datos de un cliente suministrándole al sistema únicamente su documento de identidad o código de cliente, **Para:** facilitar el proceso de búsqueda y reducir el tiempo de atención.

> **Como:** asesor de ventas, **Quiero:** disponer de una diadema inalámbrica, **Para:** poder levantarme regularmente de mi puesto y disminuir el estrés y el cansancio.

Componentes adicionales de una Historia de Usuario

Una Historia de Usuario puede tener componentes adicionales que permitan mejorar su entendimiento tanto para el usuario como para los Developers. Estos componentes también garantizan una ejecución precisa y acorde a las expectativas. Usualmente estos componentes son:

- Criterios de aceptación (detalle técnico)
- Estimaciones
- Prototipos, Diagramas, etc.

1. Criterios de aceptación (detalle técnico)

> Microsoft define a los criterios de aceptación como **las condiciones que un producto de software debe satisfacer para ser aceptado por un usuario, cliente o stakeholder.**

> Para Google, son **estándares pre-establecidos o requerimientos que un producto o proyecto debe satisfacer.**

Cada historia de usuario debe ir acompañada de sus criterios de aceptación. Estos criterios son esenciales para garantizar que todos en el equipo tengan el mismo entendimiento de lo que se espera. Ayudan a delimitar el alcance de una historia y a definir cómo se verificará que se ha cumplido con la necesidad o requerimiento descrito.

Concretamente en Scrum, se los define como un conjunto de sentencias redactadas de tal manera que conduzcan a una respuesta clara de "aceptado/rechazado". **Detallan las especificaciones técnicas de cada Historia de usuario o Tarea.**

El Product Owner es el responsable de redactar los criterios de aceptación para cada Historia de Usuario en conjunto con el cliente y/o los interesados, para posteriormente confirmarlos con los Developers antes de iniciar el desarrollo. Esta confirmación se realiza durante la Planificación del Sprint.

Los criterios de aceptación ayudan a los Developers a entender el funcionamiento del producto, de manera que estimarán mejor el trabajo necesario, además de servir como guía para tomar decisiones más acertadas de acuerdo con lo que se espera de cada Historia de Usuario.

Un criterio de aceptación debe poder probarse o testearse. Cuando no es posible probar algún criterio de aceptación es un claro indicio de una mala redacción.

Se debe tener en cuenta que al probar una Historia de Usuario frente a un criterio de aceptación se obtiene una respuesta binaria: Cumple o No cumple; **no existe el concepto de cumplir parcialmente un criterio de aceptación.**

Algunos beneficios de contar con buenos criterios de aceptación:

- Permite que los Developers entiendan una funcionalidad desde la perspectiva del usuario.
- Reduce las ambigüedades al desarrollar las Historias de Usuario.
- Promueve la calidad del producto permitiendo que los Developers se adhieran a los criterios de aceptación antes de llegar a una demostración de resultados en la Revisión del Sprint.
- Permiten confirmar si una Historia de Usuario se comporta de acuerdo con lo esperado.

Tip para el Developer

No esperes hasta la Planificación del Sprint para hacer preguntas. Si un criterio de aceptación no especifica algo, no lo asumas. Es mejor preguntar y obtener claridad.
Si durante el desarrollo encuentras que un criterio de aceptación no es viable o requiere mucho más trabajo del estimado, comunica esto al Scrum Master lo antes posible.

Una manera **opcional** de construir los criterios de aceptación es la siguiente:

Cuando (Rol) hace (Acción) consigue (Resultado / Comportamiento esperado)

Veamos un ejemplo:

> **Historia de Usuario:** *Como* cliente, *quiero* pagar mi suscripción mensual vía sitio web por medio de transferencia bancaria o tarjeta de crédito, *para* evitar el desplazamiento hasta la sucursal de pago.
>
> **Criterios de Aceptación:**
>
> - Cuando el cliente ingrese al portal de pagos del sitio web, visualiza una opción (botón) para iniciar el proceso de pago de sus suscripciones.
> - El cliente tiene la opción de seleccionar su método de pago preferido: transferencia bancaria o tarjeta de crédito.
> - Si el cliente selecciona transferencia bancaria, se le solicitan los datos bancarios y se presentan las instrucciones. Si elige tarjeta de crédito, se le solicitan los datos de la tarjeta.
> - Al ingresar la información del pago, el sistema valida que todos los campos estén completos y que los datos correspondan a formatos válidos (por ejemplo, número de tarjeta válido).
> - Una vez realizado el pago, el cliente recibe un mensaje de confirmación en la pantalla y, adicionalmente, un correo electrónico con el recibo de pago.
> - Si ocurre algún problema durante el proceso de pago (como una tarjeta declinada), el cliente recibe un mensaje claro sobre el error y las recomendaciones para solucionarlo.

Tip para el Product Owner

En la discusión de las historias de usuario, puedes emplear la técnica de las 3 C's: Card (Tarjeta), Conversation (Conversación) y Confirmation (Confirmación). Esta técnica ayuda a mantener la simplicidad, ya que la información se mantiene ligera (tarjeta), se clarifica a través del diálogo (conversación) y se verifica (confirmación).

2. Estimaciones

Las estimaciones se refieren al tiempo o esfuerzo necesario para completar una Historia de Usuario o una tarea. Los equipos Scrum a menudo usan **puntos de historia, horas o días** para estimar la

complejidad y el esfuerzo necesario.

Es muy importante realizar la estimación del trabajo requerido para desarrollar las Historias de Usuario y las tareas relacionadas, de esta manera se logran planificaciones más precisas.

La estimación **NO** es determinada por el cliente o el Product Owner, sino que son **los Developers en conjunto con el Scrum Master y el Product Owner,** quienes definen las estimaciones ya que ellos son quienes ejecutarán directamente el trabajo necesario.

□□ Tip para el Product Owner

Recuerda que las estimaciones son relativas y no deben interpretarse como compromisos rígidos. Sirven como guías basadas en la información y experiencia actual del equipo.

Para garantizar una estimación más precisa, se pueden considerar elementos como:

- Dificultad / complejidad: Hace referencia a qué tanto esfuerzo se requiere para ejecutar determinado trabajo. Se utilizan los Puntos de historia como unidad de medida.
- Duración: Hace referencia a cuánto tiempo se requiere para ejecutar determinado trabajo. Se puede utilizar horas o días como unidad de medida.
- Costo: Hace referencia al dinero requerido para la ejecución de un determinado trabajo, el cual se calcula a partir del trabajo de los developers más los recursos necesarios para su ejecución.

Uno de los problemas más comunes con la estimación de las Historias de Usuario, es sólo se estimar la dificultad (puntos de historia), o sólo estimar la duración (horas / días); por lo que es altamente recomendable estimar los dos. Muchos equipos prefieren unificar esto mediante una **relación entre los puntos de historia y la duración,** es decir que para cada punto de historia se asigna un tiempo determinado.

□□ Tip para el Scrum Master

Si sientes que las estimaciones consistentemente no reflejan el esfuerzo real de los Developers, aborda el tema en la retrospectiva del sprint. Analiza y ajusta el proceso de estimación en conjunto con el equipo según sea necesario.

El manejo que se da a los puntos de historia es **específico de cada equipo,** y es crucial entender que las estimaciones están íntimamente ligadas a la experiencia, conocimientos y dinámicas específicas de cada equipo, por lo que **NO** es recomendable comparar diferentes equipos basados en sus estimaciones: Incluso si dos equipos trabajan en el mismo producto o en productos similares, su ritmo de trabajo puede variar.

En Scrum el ritmo de trabajo también se conoce como la "velocidad" y se basa en cuántos puntos de historia, en promedio, un equipo puede completar en un Sprint. La velocidad no solo se basa en la capacidad de producción; también se ve influenciada por la comunicación dentro del equipo, las herramientas disponibles, las interrupciones externas, entre otros factores.

> **¿Por qué no comparar equipos basados en estimaciones?** Comparar equipos basados en sus puntos de historia o velocidad puede llevar a interpretaciones erróneas. Si un **equipo A completa 50 puntos** de historia en un Sprint y el **equipo B completa 30 puntos**, no significa necesariamente que el equipo A sea más productivo o eficiente. Puede que el equipo B esté manejando tareas más complejas o enfrentando desafíos que el equipo A no tiene. Comparar equipos de esta manera puede llevar a decisiones injustas y a presiones innecesarias, lo que puede afectar negativamente la moral y la productividad.

A continuación se muestra un ejemplo de ello:

Supongamos que tenemos dos equipos, Equipo A y Equipo B, ambos trabajando en diferentes funcionalidades del mismo producto.

El equipo A, con base en sus experiencias anteriores y en la comprensión que tienen sobre su capacidad y velocidad de trabajo, definió la siguiente tabla de estimaciones:

estimacion1
Image not found or type unknown

Por otro lado, el equipo B, que puede tener una composición diferente en términos de habilidades, experiencia y conocimientos técnicos, definió la siguiente tabla de estimaciones:

estimacion2
Image not found or type unknown

Durante la sesión de estimación, el Equipo A asigna 5 puntos a una historia de usuario específica, mientras que el Equipo B asigna 8 puntos a una historia similar. Esto no significa necesariamente que el Equipo B vea la historia como más complicada o que sean menos eficientes. Es posible que el Equipo B, basado en experiencias pasadas, considere ciertos riesgos o complejidades adicionales que el Equipo A no ha experimentado. Por otro lado, el Equipo A puede tener un experto en una herramienta o tecnología relevante que reduce la percepción de esfuerzo.

Si bien ambos equipos trabajan en el mismo producto, tienen distintas experiencias, habilidades y contextos que influyen directamente en cómo ven y estiman el trabajo.

□□ Tip para el Developer

Primero la calidad: No sacrifiques la calidad de tu trabajo para cumplir con una estimación. Si una historia de usuario necesita más tiempo del estimado para ser completada adecuadamente, es mejor discutir esto con el Scrum Master para ajustar el plan que entregar un producto inferior.

3. Prototipos, Diagramas, etc

Los prototipos, diagramas o esquemas de una Historia de Usuario nos sirven para mostrar un acercamiento al resultado que se espera obtener cuando los Developers ejecuten su trabajo, de esta manera se puede probar el producto antes de construirlo para identificar posibles errores y generar un mejor entendimiento para los Developers.

> **Prototipo:** Es una representación preliminar de un producto, que muestra cómo se verá y funcionará, sin necesidad de que esté completamente funcional. Puede ser desde un boceto en papel hasta una versión interactiva digital.

> **Diagrama:** Son representaciones gráficas que muestran las relaciones entre diferentes componentes, actores o flujos de un sistema o proceso.

Estos elementos visuales sirven como herramientas de comunicación entre el equipo, el Product Owner y los stakeholders, y pueden ser vitales para reducir la ambigüedad y mejorar la calidad del producto final.

Tip para el equipo

Un prototipo no tiene que ser perfecto ni estar completo: La idea es que sea una representación aproximada que pueda modificarse fácilmente en función de los comentarios recibidos.

Beneficios de usar prototipos:

- Permiten a los Developers desarrollar su trabajo en torno a una guia clara sobre las expectativas del cliente/usuario final.
- Permiten iterar y hacer cambios con rapidez antes de entrar en el desarrollo.
- Constituyen un gran apoyo para lograr mejores estimaciones de trabajo.
- Son más fáciles de abordar con los usuarios finales.
- Permite que el usuario se sienta involucrado en construcción del producto, ya que puede "verlo por adelantado".
- Se reduce el riesgo o la incertidumbre sobre el resultado de producto esperado.

Tip para el equipo

No todos los elementos o características necesitarán un prototipo o diagrama. La decisión de cuándo usarlos dependerá de la complejidad de la historia, los recursos disponibles y las necesidades del equipo y los stakeholders. Es importante recordar que, aunque estas herramientas son valiosas, deben usarse de manera estratégica para maximizar su valor sin desperdiciar esfuerzos en detalles innecesarios.

Épicas

Una épica es una gran pieza de trabajo que se puede desglosar en un conjunto de tareas más pequeñas (historias de usuario) que estan relacionadas entre sí. Puede considerarse como un **contenedor para historias de usuario que están relacionadas** y que, una vez completadas, cumplirán un objetivo o función más grande y más complejo.

Algunas ideas de las que se podrían definir las épicas son: • Módulos. • Componentes. • Hitos. • Entregables. • Funcionalidades.

> Se puede decir que una épica es un nivel de agrupación por encima de las historias de usuario que permite clasificarlas por funcionalidades, módulos, subsistemas, etc.

Las épicas:

- Son más grandes y complejas que las historias de usuario individuales.
- Se pueden descomponer en múltiples historias de usuario que, una vez completadas, satisfarán la épica en su totalidad.
- A menudo se requieren de múltiples sprints o ciclos de desarrollo para completarlas.

> El propósito de identificar épicas es ayudar a gestionar y organizar el trabajo del Product Backlog de manera más efectiva. Se trata de simplificar la complejidad y proporcionar una visión clara del objetivo final, permitiendo a los equipos trabajar de manera más coordinada y eficiente.

Recomendaciones sobre las épicas:

- Las épicas suelen **identificarse** en etapas tempranas del desarrollo, cuando el equipo se encuentra con grandes piezas de trabajo que son demasiado amplias para ser consideradas como una historia de usuario única.
- A medida que se acerca el momento de trabajar en una épica, el Product Owner en conjunto con el equipo **descompondrá la épica** en historias de usuario más pequeñas y manejables.
- Al igual que con las historias de usuario, **las épicas se priorizan.** Las épicas de alta prioridad se descompondrán antes que las épicas de baja prioridad.
- Dado que una épica se puede abarcar en múltiples sprints, se trabajará en las historias de usuario individuales que la componen a lo largo del tiempo.

> **Ejemplo de épica:** Renovar la experiencia de pago del sitio web para simplificar el proceso de compra y reducir el abandono del carrito.
>
> **Historias de Usuario derivadas de la Épica:**
>
> - *Como* cliente, *quiero* ver un resumen claro de mi pedido antes de confirmar el pago, *para* evitar errores.

> - *Como* cliente, *quiero* recibir un correo electrónico de confirmación inmediatamente después de hacer mi compra, *para* tener trazabilidad de la compra
> - *Como* cliente, *quiero* opciones de pago flexibles (tarjeta de crédito, PayPal, transferencia bancaria), *para* elegir dependiendo de mis necesidades y disponibilidad
>
> ... (y así sucesivamente, hasta que todas las facetas de la renovación de la experiencia de pago estén cubiertas).

Guía para identificar épicas sin morir en el intento ☝

- Las épicas suelen estar vinculadas a **objetivos estratégicos o de alto nivel.** Pregúntate: ¿Cuál es el gran objetivo o funcionalidad que queremos alcanzar?
- Si una historia de usuario parece **demasiado grande o ambigua,** es probable que sea una épica. Si al escribir una historia de usuario te encuentras agregando muchos **"y"** o **"o",** considera si puede ser desglosada en múltiples historias de usuario más pequeñas y, por lo tanto, convertirse en una épica.
- Si crees que una tarea o historia de usuario puede **extenderse más allá de un sprint**, es probable que estés frente a una épica.
- Discute las funcionalidades y objetivos con stakeholders y miembros del equipo. Ellos podrán proporcionar una perspectiva sobre la complejidad y el alcance, ayudando a identificar épicas.
- Si puedes desglosar un elemento en varias historias de usuario que son independientes entre sí pero que trabajan juntas para un objetivo común, entonces tienes una épica. (esto es que cada historia de usuario puede ser trabajada de forma independiente, pero que le apuntan al mismo objetivo).
- Utiliza herramientas como mapas de historias (story maps) para visualizar el flujo y las grandes áreas de funcionalidad. Estas áreas suelen ser buenas candidatas para épicas.
- Al identificar épicas, enfócate en el **"qué" (lo que se quiere lograr)** en lugar del **"cómo" (la solución técnica).** Las soluciones técnicas deben surgir al desglosar la épica en historias más pequeñas.
- A medida que el equipo avanza y se adquiere más conocimiento, es posible que algunas historias de usuario se expandan en alcance y se conviertan en épicas, o que algunas épicas se reduzcan y se conviertan en historias de usuario. Es esencial revisar y refinar regularmente.
- Si al intentar estimar una historia de usuario encuentras que el equipo tiene muchas incertidumbres o considera que llevará mucho tiempo, podría ser una señal de que estás tratando con una épica.
- Si un elemento tiene muchas dependencias o debe ser completado en un orden específico, podría ser una épica que necesita ser desglosada en historias de usuario que puedan ser trabajadas en paralelo o en un orden específico.

4.4 - Refinamiento del Product Backlog

> Estamos a puertas de iniciar con el desarrollo del producto, tenemos una comprensión general del producto y contamos con un Product Backlog que contiene elementos cortos y concretos. **Tú como Product Owner los entiendes claramente, pero: ¿el resto del equipo también los entienden?**

El refinamiento del Product Backlog es una actividad continua en la que el equipo Scrum colabora con las diferentes partes interesadas para crear un entendimiento compartido sobre lo que el producto hará y no hará (historias de usuario y tareas), sobre el esfuerzo que requerirá su implementación (estimaciones) y el orden en que lo hará (priorización).

El equipo puede llevar a cabo el refinamiento paralelamente al desarrollo del actual sprint, con esto se logrará que la planificación del siguiente sprint sea muchísimo más eficiente.

> Si bien el Product Owner lidera el proceso de refinamiento del Product Backlog, es vital que cuente con la colaboración activa de los Developers y el Scrum Master. Esta colaboración multidisciplinaria no solo enriquece el proceso con diversas perspectivas y habilidades, sino que también promueve un entendimiento compartido y una toma de decisiones más informada, lo que es esencial para el desarrollo exitoso del producto.

¿Por qué es importante el refinamiento del Product Backlog?

- Sin un entendimiento compartido, se corre el riesgo de implementar algo incorrecto, desperdiciar esfuerzos y tener que volver a trabajar en la implementación para hacerlo bien.
- Si no se dimensiona cada elemento, no se tiene en cuenta el “esfuerzo” de los elementos y se corre el riesgo de sobrevalorar los elementos de alto valor y alto coste, e infravalorar

los de menor valor y menor coste.
- Si no se ordena el Product Backlog según su prioridad, se corre el riesgo de trabajar en elementos que no son tan importantes y de pasar por alto otros importantes.

> **Caso de estudio: ¿Por qué deberíamos refinar?** En una reconocida empresa de software, la omisión del refinamiento del Product Backlog llevó a consecuencias graves. Sin una adecuada claridad y detalle en los ítems del backlog, el equipo de desarrollo frecuentemente enfrentaba obstáculos en la implementación, resultando en retrasos significativos y una acumulación de tareas mal definidas. Esto no solo generó frustración entre los miembros del equipo, sino que también impactó negativamente en la satisfacción del cliente debido a entregas de productos por debajo del estándar esperado. Este caso resalta la necesidad vital de la inclusión activa y continua del Product Owner, el Scrum Master y los Developers en el proceso de refinamiento del backlog para asegurar un flujo de trabajo fluido y entregas exitosas.

□□ Tip para el Product Owner: Mantener una lista de "ítems de refinamiento listos para discusión

Estos ítems pueden ser abordados en las sesiones con el equipo. Esto garantiza que siempre haya un flujo constante de ítems listos para ser trabajados y que las sesiones de refinamiento sean eficientes y focalizadas. Además, no dudes en retirar elementos que ya no añaden valor, manteniendo así un Backlog limpio y centrado en las prioridades actuales.

¿Qué se hace en el refinamiento del Product Backlog?

Algunas de las cosas que se puedan hacer en el refinamiento son:

- Recopilar información (puede ser mediante entrevistas, investigaciones, etc.) para redactar nuevas historias de usuario.
- Explicar el contexto de las próximas funcionalidades.
- Desglosar el trabajo (historias de usuario y tareas).
- Agregar detalles a historias de usuario ya escritas (criterios de aceptación y prototipos).
- Revisar y ajustar las prioridades de los ítems del backlog en función de la retroalimentación y aprendizajes de los sprints anteriores.
- Identificar y eliminar posibles impedimentos que puedan surgir durante la implementación de ciertas tareas, facilitando una planificación más realista y evitando futuros bloqueos.

□□ Tip para el Product Owner: No olvides involucrar a todo el equipo Scrum en el proceso de refinamiento.

Aunque como Product Owner tienes la responsabilidad de mantener y priorizar el Product Backlog, el conocimiento y la perspectiva de los Developers y del Scrum Master pueden ayudar a enriquecer los ítems del backlog, proporcionando detalles técnicos y apuntando posibles obstáculos desde una etapa temprana, facilitando una entrega más fluida y exitosa del producto.

¿Cuándo es el mejor momento para el refinamiento del Product Backlog?

El refinamiento del Product Backlog es una actividad continua, no solo para el Product Owner, sino para **todo el equipo.** El Product Owner puede refinar los elementos del Product Backlog en cualquier momento, dentro o fuera de una reunión, dependiendo de lo que sea más conveniente para el equipo.

- **En una reunión (sincrónica):** El equipo se reúne y puede invitarse a otras partes interesadas si fuese necesario. Esta reunión **NO hace parte de los eventos** de Scrum.
- **En un documento (asincrónica):** El Product Owner prepara un documento colaborativo y el resto del equipo va agregando detalles y comentarios (no se requiere reunión).

Nota: La forma y momento en que se haga el refinamiento depende del equipo Scrum.

> **Es vital destacar que la planificación en Scrum ocurre de manera paralela** Mientras los Developers están inmersos en la construcción del actual incremento de producto, el Product Owner se encuentra activamente trabajando en el refinamiento de los próximos elementos del backlog, identificando nuevas oportunidades y ajustando prioridades, garantizando así un flujo constante y eficiente de trabajo para futuros sprints.

4.5 - Técnicas y herramientas útiles para el Product Owner

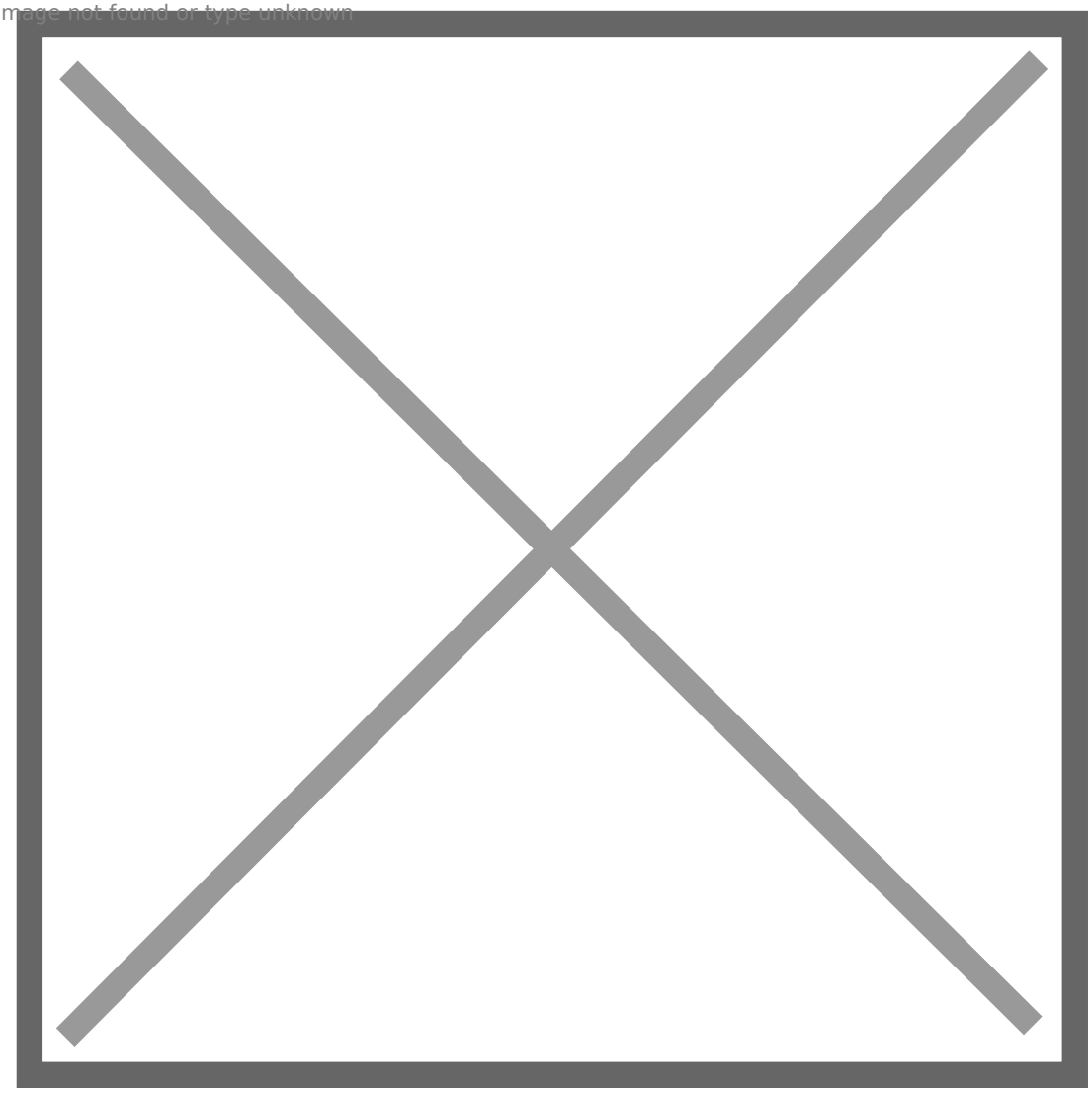
Herramienta 1: User stories mapping (Mapeo de historias de usuario)

Esta técnica es vital para organizar y visualizar de manera clara y estructurada los elementos del Product Backlog, agrupándolos según su funcionalidad y dependencia, lo que a su vez facilita su proyección a lo largo del tiempo.

El proceso de mapeo se desarrolla típicamente en sesiones de trabajo que duran entre 2 y 4 horas. Durante estas sesiones, se utiliza un lienzo y post-its para facilitar la categorización y priorización de los elementos que integrarán el producto.

Una de las grandes ventajas de este método es que asiste en la programación de los sprints, proporcionando una perspectiva clara para los stakeholders sobre qué se realizará y cuándo, aunque sea de manera aproximada.

Image not found or type unknown



> La técnica de Mapeo de Historias de Usuario (User Stories Mapping) sirve como una herramienta crucial para estructurar y definir el MVP (Producto Mínimo Viable), permitiendo una visualización clara de las funcionalidades esenciales que deben ser implementadas en primera instancia para satisfacer las necesidades centrales del cliente.

¿Cómo construir el mapa de historias?

Antes de comenzar, es fundamental establecer el estado objetivo del producto (Product Goal), ya que esto determinará el contexto dentro del cual se desplegará el producto. Posteriormente, es necesario identificar las funcionalidades o componentes principales que constituirán el producto, conocidos como "épicas".

Pasos para crear un Mapa de Historias de Usuario

1. **Definición del Product Goal:** Establece claramente el objetivo principal del producto que guiará todas las decisiones futuras, recuerda basarte en el problema que se pretende

solucionar.
2. **Identificación de Épicas:** Determina las grandes funcionalidades o componentes que conformarán el producto.
3. **Creación de Historias de Usuario:** Desarrolla historias de usuario que describan cada función desde la perspectiva del usuario.
4. **Organización de Historias:** Organiza las historias de usuario identificadas en una estructura visual, categorizándolas según su funcionalidad y dependencia.
5. **Priorización:** Establece las prioridades de las historias de usuario, decidiendo cuáles deben abordarse primero basándote en su valor y complejidad.
6. **Revisión y Ajustes:** Durante las sesiones de refinamiento del Product Backlog, revisa y ajusta el mapa según sea necesario, incorporando los aprendizajes y cambios que surjan durante el desarrollo.

Herramienta 2: Técnica para priorizar - MoSCoW (Must have, Should have, Could have, Would have)

La técnica MoSCoW es un método efectivo para **priorizar los elementos del Product Backlog**, ayudando a discernir lo esencial de lo deseable. Aquí está desglosada en sus componentes:

- **Must have (M):** Características indispensables, sin las cuales el producto no puede lanzarse.
- **Should have (S):** Características importantes que agregan valor significativo pero no son críticas para el lanzamiento.
- **Could have (C):** Características deseables pero no necesarias, pueden posponerse sin afectar gravemente el producto.
- **Would have (W):** Características que serían buenas tener en algún momento, pero no son una prioridad en este momento.

Este enfoque facilita la toma de decisiones durante el refinamiento del backlog, permitiendo una distribución más estratégica de los recursos y esfuerzos del equipo.

¿Cuándo aplicar MoSCoW en Scrum?

La técnica MoSCoW puede aplicarse en varias etapas del ciclo de Scrum:

- **Durante el Refinamiento del Product Backlog:** Para ayudar a definir y ajustar las prioridades de los elementos en el backlog.
- **En la Planificación del Sprint:** Para decidir cuáles elementos deben abordarse en el próximo Sprint, basándose en su prioridad y dependencia.

- **En las Revisiones del Sprint:** Para evaluar si las características "Must have" se han implementado correctamente y para re-priorizar características para los siguientes Sprints si es necesario.

Herramienta 3: Modelo Kano

El Modelo Kano es una herramienta útil para identificar y categorizar las necesidades del cliente en el desarrollo del producto. Se clasifica en las siguientes categorías:

- **Características Básicas:** Elementos que los clientes dan por sentado. Su ausencia causaría insatisfacción, pero su presencia no aumenta la satisfacción.
- **Características de Rendimiento:** Aspectos que pueden incrementar la satisfacción del cliente según su grado de implementación.
- **Características de Excitación:** Innovaciones que los clientes no esperan, pero que podrían deleitarlos y generar una ventaja competitiva.

Al aplicar el Modelo Kano, el Product Owner puede obtener una comprensión más profunda de lo que realmente desea el cliente, lo que puede ayudar a priorizar de manera efectiva las funcionalidades en el Product Backlog.

Herramienta 4: Customer Journey Maps (Mapas del Viaje del Cliente)

El Customer Journey Map es una herramienta visual que representa la experiencia total del cliente con un producto o servicio desde el primer contacto hasta la interacción final. Ayuda a comprender las necesidades, las emociones y los obstáculos que el cliente podría enfrentar en cada etapa. Aquí está su estructura general:

- **Puntos de contacto:** Donde y cuando el cliente interactúa con el producto o servicio.
- **Motivaciones:** ¿Por qué el cliente se siente motivado para continuar o abandonar?
- **Preguntas:** Dudas o inquietudes que el cliente pueda tener en cada etapa.
- **Emociones:** Cómo se siente el cliente en cada punto de contacto.

El uso de los Customer Journey Maps permite al Product Owner:

- Identificar áreas de mejora en la experiencia del cliente.
- Priorizar características basadas en momentos críticos del viaje del cliente.
- Diseñar soluciones que se alineen mejor con las expectativas y necesidades del cliente.

Otras herramientas / técnicas útiles para el Product Owner

- **Técnica de Estimación Poker (explicada en la sección de Developer):** Utilizada para estimar el esfuerzo requerido para cada tarea.
- **Impact Mapping:** Una técnica estratégica de planificación que visualiza las metas y el impacto esperado de un proyecto.
- **Herramientas de Gestión de Proyectos Ágiles (como JIRA, Trello, Asana):** Facilitan la organización y seguimiento de las tareas y progresos del equipo.
- **Lean Canvas:** Una herramienta que ayuda a desarrollar y afinar el modelo de negocio de un producto.
- **Analytics Tools:** Herramientas como Google Analytics para entender mejor el comportamiento del usuario y tomar decisiones informadas.

---

5. Developer: Desarrollo ágil de productos

Los Developers tienen la responsabilidad vital de dar vida a los incrementos generados en cada Sprint, construyendo los elementos que eventualmente conformarán el producto final. Es imperativo que posean las habilidades y competencias necesarias para realizar este trabajo de manera efectiva. En este capítulo, nos sumergiremos en el mundo de los Developers, destacando las actividades, habilidades y responsabilidades esenciales que delinean su función dentro del equipo Scrum, complementando lo ya discutido en el modelo core de Scrum. Puntos clave de este capítulo: - Identificar las responsabilidades centrales de los Developers en el proceso Scrum. - Explorar las habilidades y competencias que los Developers deben cultivar. - Analizar las actividades diarias y estrategias que facilitan un desarrollo de producto exitoso. - Comprender cómo los Developers colaboran para alcanzar los objetivos de los Sprint y contribuir al producto final.

5.1 - Desarrollo ágil de producto

En el dinámico mundo actual, la figura del Developer se transforma constantemente, abrazando roles más estratégicos y creativos. En este escenario, el desarrollo ágil de productos emerge como un enfoque que no sólo refina tus habilidades técnicas, sino que potencia tu capacidad para contribuir activamente en la creación de soluciones innovadoras.

En este módulo, te invitamos a explorar cómo, como developer, puedes ser un actor clave en este proceso. Desde participar activamente en la definición de la arquitectura del producto, hasta adoptar prácticas de documentación eficientes y ser un protagonista en la implementación de pruebas continuas, tu responsabilidad es esencial para el éxito del producto.

Prepárate para sumergirte en un camino donde tu conocimiento técnico se une con una visión estratégica, permitiéndote participar de manera significativa en la entrega continua de valor. ¡Adelante!

5.1.1 - Autogestión y Gestión del Tiempo: Claves en un buen Scrum Developer

Dentro de un equipo Scrum, se espera que los Developers sean proactivos y asuman un papel central en la autogestión de su tiempo y esfuerzos. Esto no solo fomenta la responsabilidad individual, sino que también promueve una cultura de colaboración y respeto mutuo.

> **Caso práctico:** Imagina un escenario donde un Developer, Carlos, se da cuenta de que una tarea específica está consumiendo más tiempo del esperado, lo cual puede afectar la entrega oportuna del incremento planeado para el Sprint. En lugar de esperar a ser dirigido, toma la iniciativa de comunicar la situación al equipo, proponiendo una sesión de brain-storming para encontrar una solución eficiente. Gracias a esta proactividad, el equipo pudo reajustar y repriorizar las tareas, garantizando la entrega exitosa del incremento.

Tips para una Efectiva Autogestión y Gestión del Tiempo

- **Establece Metas Claras:** Asegúrate de entender bien el objetivo de cada Sprint y cómo contribuyen tus tareas a alcanzarlo.
- **Comunicación Continua:** Fomenta la comunicación constante con tu equipo para identificar y abordar problemas a tiempo.
- **Priorización Efectiva:** Aprende a priorizar tareas basándote en su valor y urgencia, esto ayudará a evitar el desperdicio de tiempo en elementos menos críticos.
- **Feedback Continuo:** Promueve una cultura de feedback constante para facilitar la mejora continua y la adaptación ágil.

> La autogestión, combinada con una efectiva gestión del tiempo y priorización, no solo facilita la entrega oportuna de productos de alta calidad, sino que también contribuye a un ambiente de trabajo más armonioso y productivo.

5.1.2 - Consideraciones clave para el desarrollo ágil de productos

En el trayecto de construcción de productos ágiles, es imperativo tener en cuenta varios factores esenciales que influenciarán el éxito del producto final. Aquí te presentamos tres preguntas centrales que todo Developer debería tener en mente:

- ¿Por qué debería definirse la arquitectura del producto?
- ¿Qué debo considerar respecto a la documentación del producto?
- ¿Qué implican exactamente las pruebas del producto?

¿Por qué debería definirse la arquitectura de producto

> La "Arquitectura del Producto" puede definirse como el esquema estructural que ilustra cómo se organizarán y interactuarán las distintas partes o componentes de un producto. Esta estructura no solo comprende la organización de elementos técnicos, sino también la definición de las relaciones y dependencias que existen entre ellos. El establecimiento de una arquitectura sólida desde el principio, ayuda a prever desafíos y facilita el proceso de incorporación de cambios o ajustes en etapas posteriores del desarrollo. En pocas palabras, funciona como el mapa guía que orienta el desarrollo del producto de una manera organizada y estratégica, asegurando que todas las piezas encajen de manera óptima para cumplir con los objetivos propuestos.

Definir la arquitectura del producto no es solo una cuestión de organización, sino que establece una hoja de ruta clara que facilita la identificación y solución de posibles problemas antes de que surjan. Permite una colaboración fluida entre los miembros del equipo y garantiza que todos trabajen hacia una visión unificada, optimizando los recursos y el tiempo disponible.

> Antes de iniciar la construcción del producto es importante identificar la relación entre todos los componentes que harán parte del mismo. Definir una correcta arquitectura permitirá que los equipos de trabajo sepan como organizar sus entregables, se facilitará la integración y la mejora continua.

Es indispensable que, antes de iniciar con el desarrollo se defina el diseño técnico del producto o del incremento de producto a desarrollar. Esta actividad podría hacerse de forma iterativa antes de desarrollar los incrementos en los distintos Sprint, o de forma global al inicio del desarrollo.

Para garantizar que la arquitectura de producto cumple exactamente con los requerimientos y necesidades del cliente se debe hacer en conjunto entre los Developers y el Product Owner.

Tip para el Developer

Involúcrate activamente en las discusiones sobre la arquitectura del producto. Tu conocimiento técnico es vital para diseñar una arquitectura que no solo cumpla con los requisitos del cliente, sino que también sea sostenible y escalable a largo plazo. No dudes en compartir tus ideas y sugerencias.

> La arquitectura está estrechamente relacionada con el objetivo del producto (Product Goal). El diseño arquitectónico del producto o sistema sienta las bases para lograr los objetivos y metas del producto, asegurando que se puedan implementar las características y funcionalidades deseadas de manera efectiva.

Ejemplo de arquitectura 1: Construyendo un vehiculo electrico

Al embarcarse en la construcción de un vehículo eléctrico, es vital analizar cuidadosamente los siguientes aspectos cruciales y entender cómo interactuarán entre sí y quién será responsable de su desarrollo, algunos ejemplos que debes considerar al definir la arquitectura son:

- **Sistema de Propulsión:** ¿Motor eléctrico de eje central o trasero? ¿batería de iones de litio o plomo? (en el piso del auto).
- **Sistema de Frenado:** ¿Frenos regenerativos para recuperar energía o solo ABS tradicional?.
- **Sistema de Navegación:** ¿GPS integrado con asistentes de conducción autónoma?.
- **Interfaz de Usuario:** ¿Tablero digital o analógico? ¿Controles táctiles o de voz?.
- **Conectividad:** ¿Bluetooth, Wi-Fi? ¿Conectividad celular para actualizaciones y telemetría?.

Ejemplo de arquitectura 2: Construyendo un software LMS

Una empresa de desarrollo de software planea construir una aplicación para realizar cursos virtuales. Al ser un proyecto de software, en la definición de arquitectura se deberán tener en cuenta los siguientes elementos:

- **Front-end:** ¿Será HTML, CSS, JavaScript para la interfaz de usuario?.
- **Back-end:** ¿Se usará un framework de base Ej. Laravel PHP para la lógica del servidor??
- **Base de datos:** ¿Qué motor de base de datos? ¿Relacional o no relacional?
- **APIs:** ¿JSON Web Token para autenticación? RESTful APIs para interactuar con servicios externos, como plataformas de redes sociales o bases de datos de terceros.
- **Infraestructura:** ¿Qué proveedor? ¿Qué servicios? ¿Cómo será la red? Servidores en la nube con escalabilidad automática.

Ejemplo de arquitectura 3: Construcción: Diseño de una Casa Moderna

- **Cimientos:** ¿Será hormigón armado o pilotes?¿Madera?
- **Estructura Principal:** ¿Marco de acero o madera para las paredes y techos?
- **Sistema de Techo:** ¿Tejas de asfalto o metal?
- **Sistema Eléctrico:** ¿Cableado empotrado o visible? (puntos de luz, enchufes y switches).
- **Plomería:** ¿Tubos de PVC o cobre? (suministro de agua y desagüe).
- **Espacio Habitacional:** Distribución de habitaciones, baños, cocina, etc.
- **Sistema de Climatización:** ¿Aire acondicionado centralizado o radiadores?
- **Energía eléctrica:** ¿Se usará energía alternativa en el futuro? ¿Se van a instalar múltiples medidores?

¿Qué debo considerar respecto a la documentación del producto?

La documentación es una herramienta vital en el desarrollo de productos, ya que facilita la trazabilidad y comprensión de las funcionalidades y características del producto. Es importante considerar no solo qué documentar, sino cómo hacerlo de una manera que sea accesible y útil para todos los involucrados. Además, incentivar la documentación proactiva puede ser un método efectivo para garantizar que se mantenga actualizada y relevante.

La documentación es una herramienta indispensable que, aunque a veces despreciada, puede ser un salvavidas en el ciclo de vida del desarrollo de productos. Vamos a desglosar cómo puede realizarse de manera eficiente y sin dolor:

- Muchos trabajadores odian documentar.
- Se pueden definir incentivos por mantener una buena documentación (no necesariamente económicos).
- Definir las reglas detalladas de qué se debe documentar y cómo.
- Se puede crear una Wiki y allí almacenar todo el contenido de forma organizada.

- Se puede incluir la documentación como parte de los criterios de la Definición de Terminado (DoD).

Tip para el Developer

Considera la documentación como una herramienta que te ayudará, no como una tarea tediosa. Al mantener una documentación precisa y bien organizada, facilitas la gestión de la calidad y la colaboración efectiva en tu equipo.

> **Caso práctico** Pedro, uno de los developers, toma la iniciativa de crear una Wiki para almacenar toda la documentación del proyecto. A medida que el proyecto avanza, el equipo la mantiene actualizada, facilitando la incorporación de nuevos miembros al equipo y sirviendo como una valiosa referencia para revisar decisiones pasadas.

> **Caso Práctico: Motivando la Documentación** Laura, una developer experimentada, notó que el equipo evitaba la documentación, lo que causaba confusión y retrasos. Para remediarlo, instauró junto al Scrum Master sesiones dedicadas a la documentación en cada sprint, complementándolo con un sistema de recompensas para el miembro que más contribuyera. Este enfoque transformó la documentación de una tarea despreciada a una actividad valorada que mejoraba la eficiencia del equipo.

¿Qué implican exactamente las pruebas del producto?

Las pruebas no son una fase aislada en el proceso ágil, sino una integración continua que asegura la calidad y el valor del producto en cada etapa del desarrollo. Los Developers deben adoptar una postura proactiva hacia la realización de pruebas, asegurando que cada incremento cumpla con los estándares de calidad establecidos y contribuyendo al éxito a largo plazo del producto.

> **Una prueba es la forma de comprobar el correcto funcionamiento del producto o uno de sus incrementos.**

Tip para el Developer

Adopta un enfoque proactivo hacia las pruebas, viéndolo como una oportunidad para mejorar continuamente el producto, en lugar de una tarea de última hora. Incorpora pruebas regulares en tu flujo de trabajo para garantizar que cada incremento del producto sea de alta calidad.

Tip para el Developer

Recuerda que las pruebas no son una tarea aislada o exclusiva de un equipo de pruebas; eres co-responsable de la calidad de los entregables. Aprovecha tu conocimiento y habilidades para integrar pruebas continuas. Considera que las pruebas son exclusivas para proyectos de software, sino en todas las facetas del desarrollo de productos. Esto asegurará que el producto no solo funcione como se espera, sino que también cumpla con los estándares de calidad desde el inicio.

- En agilidad no se consideran a las pruebas como una fase separada, sino como parte integral del desarrollo de producto.
- En metodologías tradicionales, es una fase más dentro del desarrollo cascada o waterfall, lo cual no siempre resulta eficiente.
- Desde la agilidad se prueba continuamente, porque es una de las claves para mantener la calidad del producto.

> **Caso práctico: Cuando no se hacen pruebas o no son suficientes** En un equipo, inicialmente se pasaba por alto la fase de pruebas o se realizaban de manera muy superficial, lo que llevaba a múltiples errores en las versiones finales del producto. Pero, al establecer definiciones de terminado (DoD) centradas en procesos específicos, se creó una pauta clara para realizar pruebas exhaustivas en cada incremento. Esto no solo redujo los errores sino que también facilitó que cada miembro del equipo pudiera llevar a cabo pruebas efectivas de sus propios entregables, mejorando así la calidad general del producto.

5.2 - Los Developers y los eventos

El desarrollo de un producto exitoso es una jornada colaborativa, y los Developers están en el corazón de este viaje. En el contexto de Scrum, su participación activa en los diversos eventos es crucial, no solo por su papel en la construcción del producto sino también por su contribución a la planificación, adaptación y mejora continua que Scrum fomenta.

En esta sección, desglosaremos cómo los Developers se involucran y contribuyen en cada uno de los eventos de Scrum. Desde la planificación del Sprint, donde establecen las bases del trabajo a realizar, hasta la retrospectiva del Sprint, donde reflexionan sobre el progreso y buscan formas de mejorar. A través de su compromiso y colaboración en cada fase, garantizan que el producto en desarrollo esté alineado con los objetivos establecidos y se adapte eficazmente a las cambiantes demandas y expectativas.

◫◫ Tip para los Developers

Durante los eventos de Scrum, toma un papel activo en la comunicación y colaboración. No sólo compartas el progreso técnico, sino también busca feedback y aporta ideas para potenciar el valor del producto en desarrollo. La colaboración efectiva entre Developers y otras responsabilidades en el equipo Scrum puede ser la clave para la creación de soluciones innovadoras y de alto impacto.

Los Developers en la Planificación del Sprint

- Participan del evento, identificando los 3 objetivos: ¿por qué este Sprint es valioso?, ¿qué se puede terminar (done) en este Sprint?, ¿cómo se realizará el trabajo elegido?
- Indagan detalladamente sobre el trabajo que se pueda terminar este Sprint
- Aclaran los aspectos técnicos a las partes interesadas para lograr un entendimiento compartido del trabajo del Sprint
- Estiman el trabajo seleccionado para el Sprint de acuerdo a su experiencia y conocimiento técnico.

> **Caso práctico:** Durante la planificación del Sprint, el equipo se reúne para discutir las metas del próximo Sprint. Daniela, una Developer, sugiere que se enfoquen en un conjunto particular de historias de usuario que, según su análisis, pueden agregar un valor significativo al producto. Aprovecha este momento para clarificar dudas técnicas con el Product Owner y garantiza que

> todos los elementos están bien definidos. Juntos, como equipo, realizan una sesión de estimación basada en su experiencia y conocimientos técnicos, asegurando una comprensión compartida de la carga de trabajo y de los objetivos del Sprint.

□□ Tip para los Developers

En la planificación del Sprint, no te limites a hacer caso a lo propuesto por el Product Owner, adopta una postura proactiva en identificar las historias de usuario que podrían agregar más valor al producto. Utiliza este tiempo no sólo para entender el "qué" sino también el "por qué" detrás de cada elemento del Product Backlog, alineándolos estratégicamente con el Objetivo del Sprint (Sprint Goal) y el Objetivo de Producto (Product Goal) a largo plazo.

Los Developers en el Daily

- Llevan a cabo el evento, cumplen con la agenda y respetan su duración
- Reportan los impedimentos que puedan encontrar en el trabajo diario del Sprint
- Realizan el Daily a la misma hora y en el mismo lugar
- Informan al resto del equipo sobre su progreso diario

> **Consejo:** Recuerda que el objetivo del Daily no es supervisar sino facilitar la colaboración y el flujo de información. Anima a los Developers a que se apropien de este espacio, compartiendo no sólo avances sino también aprendizajes y retos, promoviendo una cultura de transparencia y confianza mutua, y evitando la percepción de que se trata de una herramienta de supervisión.

□□ Tip para los Developers

Evita la Monotonía: Para prevenir que los Daily se conviertan en encuentros monótonos, varía la estructura ocasionalmente. Por ejemplo, puedes introducir una breve sesión de brainstorming para solucionar un problema específico o facilitar una rápida actividad de team-building que fomente la colaboración y la energía positiva en el equipo.

Los Developers durante el desarrollo del Sprint

- Desarrollan los elementos que fueron identificados en la Planificación del Sprint
- Cumplen con los criterios de aceptación y definición de terminado al desarrollar su trabajo
- Participan del refinamiento de los elementos del Product Backlog a partir de las sesiones que gestione el Product Owner con el cliente y partes interesadas
- Se coordinan para abordar las dependencias que se puedan presentar en el Sprint

Consejos para Developers durante el Desarrollo de un Sprint

- **Colaboración Continua:** Fomenta una cultura de colaboración y apoyo mutuo. No dudes en pedir ayuda o brindar asistencia a tus compañeros de equipo si ves que pueden beneficiarse de tu experiencia o conocimientos.

- **Mantenimiento de la Calidad:** Dedica tiempo a revisar y mejorar la calidad de los incrementos en los que trabajas. Aprovecha las prácticas como la revisión con tus colegas para asegurar que el producto mantenga un estándar de calidad elevado.
- **Gestión Efectiva del Tiempo:** Desarrolla una estrategia para gestionar tu tiempo eficientemente durante el Sprint. Esto podría incluir la utilización de técnicas como la técnica Pomodoro para mantener un enfoque sostenible y evitar el agotamiento.

> **Caso Práctico:** Durante el desarrollo del Sprint, Laura, una Developer, se da cuenta de que una funcionalidad que están implementando podría beneficiarse de una integración con una herramienta externa, lo que potencialmente podría ahorrar tiempo y esfuerzo en el futuro. **Acción:** Laura discute esta idea con el resto del equipo durante una sesión de trabajo. Juntos, evalúan los pros y los contras de la implementación de esta integración en este momento, teniendo en cuenta el Objetivo del Sprint y las capacidades del equipo. **Resultado:** Después de una discusión constructiva, deciden adaptar el plan de Sprint para incorporar esta mejora, ajustando las historias de usuario y las tareas asociadas según sea necesario. Esto les permite entregar un incremento de mayor valor al final del Sprint, demostrando una adaptabilidad y una perspectiva centrada en el valor.

Los Developers en la Revisión del Sprint

- Realizan la presentación del incremento de producto logrado durante el Sprint
- Discuten sobre el progreso respecto al Objetivo de Producto
- Discuten el Product Backlog tal y como está
- Generan compromisos frente al trabajo que no se terminó

Consejos para Developers durante la Revisión del Sprint

- **Preparación de Demostraciones:** Asegúrate de que las demostraciones estén bien preparadas y enfocadas en mostrar el valor añadido durante el Sprint, en vez de solo mostrar características técnicas (a veces los interesados no piensan tanto en las características técnicas si no es como su problema o necesidad es resuelta).
- **Feedback Constructivo:** Fomenta una cultura de feedback constructivo, donde se valoren las opiniones de todos los participantes, incluyendo a los stakeholders.
- **Identificación de Mejoras:** Aprovecha la revisión del Sprint para identificar áreas donde el equipo puede mejorar en futuros Sprints, manteniendo una mentalidad de mejora continua.

> **Caso práctico:** En una sesión de revisión del Sprint, los Developers se encuentran presentando el incremento de producto que lograron durante el Sprint. Laura, una de las Developers, toma la iniciativa de explicar cómo abordaron ciertos retos técnicos y lograron cumplir con los criterios de aceptación establecidos. A medida que se presentan los elementos, Juan, otro

> Developer, muestra ejemplos prácticos de las funcionalidades implementadas, permitiendo que las partes interesadas tengan una comprensión más clara del progreso. Juntos, los Developers colaboran para asegurar que todas las voces sean escuchadas y que el feedback recibido sea adecuadamente integrado en las consideraciones para el próximo Sprint.

Los Developers en la Retrospectiva del Sprint

- Participan en la inspección de cómo fue el último Sprint con respecto a los individuos, las interacciones, los procesos, las herramientas, y su Definición de terminado (DoD).
- Identifican las lecciones aprendidas y las suposiciones que los llevaron por el mal camino, lo que fue bien durante el Sprint, los problemas que encontraron, y cómo esos problemas fueron (o no fueron) resueltos
- Discuten y reflexionan sobre la velocidad del equipo en el Sprint

> **Aclaración:** Una retrospectiva no solo incluye la identificación de problemas, sino también la colaboración proactiva para diseñar soluciones viables y establecer metas claras para mejorar en áreas específicas.

□□ Tip para el Developer en la Retrospectiva

Cambia el Entorno y Añade un Toque Personal: No permitas que las retrospectivas se conviertan en una rutina aburrida. Sugiere realizar estas sesiones en una herramienta de colaboración gráfica, donde el equipo pueda visualizar de manera creativa los puntos de mejora, o incluso fuera de la oficina en un ambiente más relajado. Además, ¿por qué no agregar un elemento de diversión? Podría ser organizar una comida especial o un snack diferente para cada retrospectiva, fomentando un ambiente más ameno y relajado que puede ayudar a que fluyan mejor las ideas y la comunicación.

Consejos para Developers durante la Retrospectiva del Sprint

- **Valora la Retrospectiva:** La retrospectiva no es una mera formalidad, es una oportunidad crucial para reflexionar y crecer como equipo. Es importante que cada miembro del equipo entienda el valor de estas sesiones y participe activamente en ellas.
- **Abordar Temas Incomodos:** No evites los temas incomodos. Es vital abordar cualquier problema o desafío que el equipo pueda estar enfrentando para fomentar una mejora continua.
- **Fomenta la Comunicación Abierta:** Fomenta una atmósfera donde todos se sientan cómodos para compartir sus opiniones y feedback, esto podría llevar a descubrir áreas de mejora que no eran obvias en primera instancia.

> **Caso práctico:** En la retrospectiva, algunos miembros del equipo mencionan que sienten que no hay mucho que discutir o mejorar, mientras que otros sienten que hay problemas pero son reacios a hablar de ellos para evitar

> situaciones incómodas. **Acción:** Laura, una de las developers, decide tomar la iniciativa. Ella comparte honestamente algunos desafíos que ha notado durante el Sprint, aunque son temas sensibles. Su apertura anima a otros a compartir sus propias observaciones y preocupaciones, lo que lleva a una discusión enriquecedora. **Resultado:** El equipo logra identificar varios aspectos cruciales para trabajar y mejorar en el próximo Sprint. Además, se dan cuenta de que, enfrentando los temas incomodos, pueden encontrar soluciones conjuntas y fortalecer la dinámica del equipo.

5.3 - Técnicas y herramientas útiles para los Developers

En el dinámico mundo de Scrum, las responsabilidades de los Developers trascienden las fronteras de las tareas convencionales. Se espera que estén equipados para abordar una amplia variedad de desafíos, adaptándose con agilidad a las demandas cambiantes del proyecto. En esta sección, exploraremos algunas técnicas y herramientas cruciales que pueden facilitar a los Developers esta navegación a través del vibrante entorno de Scrum. Dentro de ellas veremos:

- Técnicas para el desarrollo ágil
- Técnicas para estimar el trabajo
- Técnicas y herramientas de software para trabajar con Scrum

> **Consejo** Al seleccionar y emplear diversas técnicas y herramientas, es vital hacerlo con un enfoque centrado en el valor y la eficiencia. Considera la compatibilidad con las necesidades del organización, la facilidad de integración y cómo pueden potenciar una colaboración efectiva y una entrega ágil de incrementos de alta calidad. A menudo, una herramienta simple pero bien elegida o una técnica aplicada con pericia puede marcar una significativa diferencia, optimizando procesos y fomentando un ambiente de trabajo más productivo y armonioso.

5.3.1 - Para el desarrollo ágil

En esta sección, exploraremos técnicas y herramientas que pueden facilitar un proceso de desarrollo más fluido y colaborativo, promoviendo la calidad y eficiencia en cada incremento.

> **Nota:** Para aquellos interesados específicamente en el ámbito del desarrollo de software, les invitamos a consultar nuestra otra publicación enfocada especialmente en este sector, disponible a través de CertMind, donde

> profundizamos en técnicas y herramientas adaptadas a las particularidades del desarrollo de software.

Programación por parejas – Pair programming

La programación por parejas es una técnica de desarrollo ágil en la que dos desarrolladores colaboran en la misma tarea, trabajando en un único equipo o computadora. Este enfoque fomenta la colaboración, la comunicación directa y un intercambio fluido de ideas, lo que a menudo resulta en soluciones más innovadoras y eficientes.

Aunque puede parecer que esta técnica reduce la cantidad de funcionalidad producida, en realidad, tiende a mejorar notablemente la calidad del resultado final. Al tener dos pares de ojos revisando el trabajo en tiempo real, es más probable que se identifiquen y corrijan errores en una etapa temprana, lo que puede prevenir costos adicionales a largo plazo.

- En la programación por parejas, 2 Developers trabajan en un solo PC.
- La programación por parejas mejora notablemente la calidad del incremento, sin impactar el tiempo de entrega.
- Una sola persona sola agrega más funcionalidad, pero con menos calidad.
- La calidad alta puede ahorrar costes futuros en un proyecto.

□□Tip para el Pair Programming

La programación por parejas no es una sesión de enseñanza; de hecho, funciona mejor cuando ambos desarrolladores tienen un nivel similar de conocimiento y habilidades. Esto permite una colaboración más fluida y equitativa, donde ambos pueden contribuir significativamente al progreso de la tarea. Asegúrate de emparejar a desarrolladores con niveles de habilidad compatibles para aprovechar al máximo esta técnica.

El pair programming no solo se usa en programación de software

- **Diseño Gráfico:** Dos diseñadores podrían trabajar juntos en un diseño, con uno manejando las herramientas de diseño y el otro proporcionando retroalimentación en tiempo real.
- **Redacción y Edición:** Un escritor y un editor podrían colaborar en tiempo real en un documento. Uno escribe mientras el otro revisa la gramática, el estilo, etc.
- **Análisis de Datos:** En un proyecto de análisis de datos, un estadístico podría implementar modelos mientras otro experto en el dominio proporciona contexto y verifica la validez del análisis.
- **Educación:** En un entorno educativo, dos instructores podrían colaborar para enseñar una clase en línea o preparar material didáctico.
- **Investigación Científica:** Dos investigadores podrían colaborar en un experimento o en la redacción de un artículo científico.
- **Planeación Estratégica en Negocios:** En la gestión y planeación estratégica, un par podría trabajar en el análisis SWOT de un proyecto o en la elaboración de un plan de negocios.

- **Gestión de Proyectos:** Un administrador de proyecto y un stakeholder podrían revisar juntos el progreso de un proyecto, identificando riesgos y planificando mitigaciones en tiempo real.

> **Caso práctico: El pair programming puede ser agotador** En una empresa de tecnología, se instauró la técnica de programación por parejas como una actividad regular pero no constante. Los equipos encontraron que trabajar en parejas durante periodos específicos (como por la mañana o en sesiones de no más de dos horas) era beneficioso sin ser agotador. Esta moderación permitió mantener altos niveles de energía y concentración, favoreciendo la calidad y eficiencia del trabajo realizado.

> **Caso práctico: Cambiando de pareja** En una startup, se estableció una rotación de parejas de programación con cada nuevo Sprint. Esta rotación no solo permitió que los Developers fortalecieran sus relaciones interpersonales, sino que también promovió una comprensión más profunda y diversificada del proyecto en su conjunto, ya que cada miembro del equipo tuvo la oportunidad de trabajar y aprender de diferentes perspectivas y enfoques a lo largo del tiempo.

□□Tip: el Pair Programming no se aplica a todas las tareas

El "Pair Programming" no es una regla fija que deba aplicarse a todas las tareas. Es una herramienta especialmente efectiva para abordar desafíos complejos, explorar nuevas tecnologías o desarrollar funcionalidades críticas. Considera implementarlo estratégicamente, especialmente en situaciones donde la colaboración directa puede potenciar la creatividad, la calidad y la eficiencia. Recuerda, la clave es usarlo donde agregue el mayor valor al proceso de desarrollo del producto.

La refactorización

La refactorización, en un contexto más amplio que el desarrollo de software, puede ser entendida como el proceso de revisar, ajustar y mejorar las estructuras existentes de un proyecto o sistema para hacerlo más eficiente, manejable o adaptable a nuevas necesidades, sin alterar su función fundamental. Esto puede aplicarse a una variedad de ámbitos, como procesos empresariales, estrategias de marketing, estructuras organizativas, etc.

> **Ejemplo: Antes de la Refactorización** Un equipo utiliza una plantilla de informe densa y complicada, que contiene una cantidad significativa de campos irrelevantes. Esto no solo consume tiempo sino que también dificulta la comprensión clara de los puntos clave. **Después de la Refactorización** - El equipo decide revisar y simplificar la plantilla, eliminando los campos

> innecesarios y dando prioridad a la información más crítica. Como resultado, los informes ahora son más concisos, centrados y fáciles de leer, facilitando la toma de decisiones y la comunicación efectiva dentro del equipo.

Tip: La refactorización debe ser vista como una actividad continua

La refactorización es un compromiso con la mejora constante. No es un "arreglo único", sino un proceso en el que se está siempre atento a las oportunidades de optimizar y ajustar para alcanzar una mayor eficiencia y efectividad. Además, es vital evitar la "refactorización por refactorización". Cada cambio debe tener un propósito claro y estar orientado a lograr mejoras tangibles, evitando alteraciones que no agregan valor significativo.

5.3.2 - Para estimar el trabajo

La estimación consiste en definir entre todos una suposición lo más precisa posible, de lo que se puede lograr y en cuanto tiempo.

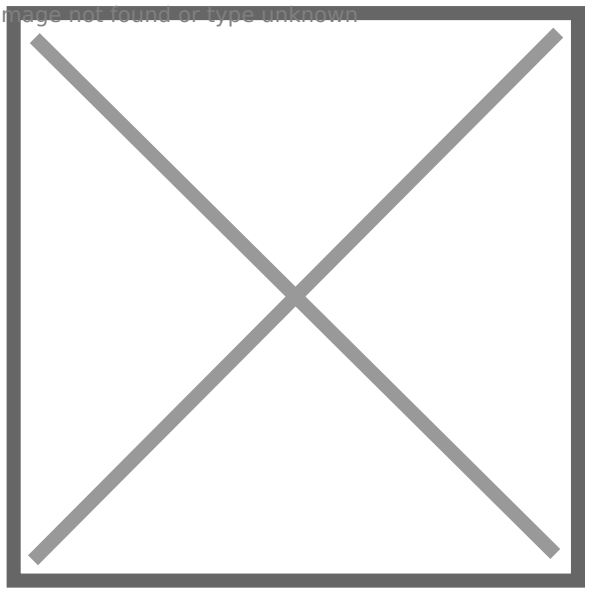
La estimación permite:

- Analizar dependencias
- Organizar las actividades prioritarias
- Definir las mejores técnicas de trabajo para ese Sprint
- Pronosticar diferentes escenarios (predecir tiempos y fechas de entrega)

Nota: El momento en que se realiza la estimación es durante la Planificación del Sprint.

Estimar con Planning Poker

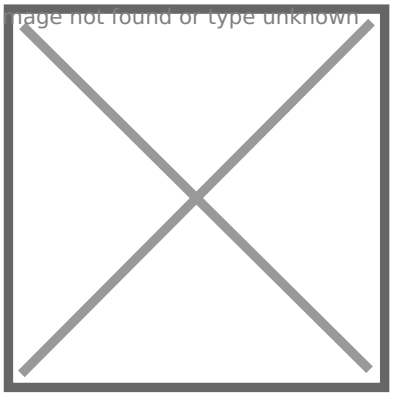
Image not found or type unknown



El póker de planificación consiste en un conjunto de tarjetas numéricas que sirven para realizar la estimación de tareas o historias de usuario.

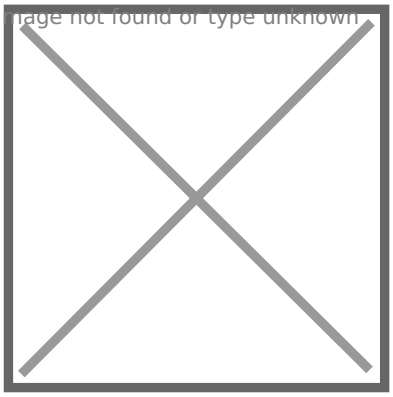
El póker puede ir numerado de 1 a 10 o utilizando la sucesión de Fibonacci

Estimar por afinidad



Consiste en encontrar las similitudes de las Historias de Usuario y tareas que se van a estimar. Lo que hace el equipo es agrupar los elementos que son similares. La mejor manera de “encontrar similitudes” es visualizar el panorama completo y luego agrupar elementos.

Estimar con Tallas de camiseta (t-shirt)



Se asigna una talla de camiseta a cada Historia de Usuario o tarea (desde XS hasta XXL) que representará el esfuerzo relativo de ese trabajo.

Con esta técnica se elimina la necesidad de la precisión de un modelo numérico, por lo que el equipo puede pensar de manera más abstracta en las estimaciones del trabajo a estimar.

¿Qué pasa cuando se Subestima/Sobreestima un Sprint?

Cuando los equipos Scrum son nuevos, o tienen poca experiencia es típico que sucedan las siguientes 2 situaciones:

Sobreestimación: Cuando los Developers proponen demasiado tiempo para el trabajo del Sprint y acaban mucho antes de la Revisión del Sprint. En este caso el Product Owner agregará más trabajo

al Sprint en curso (para que la Revisión no se cambie de fecha).

Subestimación: Cuando los Developers proponen muy poco tiempo para el trabajo del Sprint, lo que puede causar que el incremento no esté completamente terminado. En este caso la Revisión se hace sobre los elementos que se hayan completado y los demás quedan pendientes para ser terminados en el próximo Sprint (la revisión NO se aplaza).

Otras posibles técnicas que podrían ser útiles para mejorar el flujo de trabajo durante un Sprint

- Mesa de ayuda cruzada (Cross-Functional Help Desk): Un espacio donde los miembros del equipo pueden ofrecer o solicitar ayuda en diferentes áreas de especialización, fomentando la colaboración y el aprendizaje mutuo.
- Time-boxing: Establecer un tiempo límite para ciertas actividades puede ayudar a mantener el enfoque y evitar la procrastinación, así como fomentar la toma de decisiones eficiente.
- Revisión intermedia (Mid-Sprint Review): Una revisión a mitad de camino del Sprint puede ayudar a identificar cualquier obstáculo o cambio necesario a tiempo, permitiendo ajustes antes de la Revisión de Sprint.
- Kanban visual: Utilizar un tablero visual Kanban para monitorear el progreso de las tareas, lo que puede facilitar la identificación de cuellos de botella y ayudar a mantener un flujo de trabajo estable.
- Daily Stand-ups Mejoradas: Incorporar segmentos breves donde se compartan conocimientos o trucos útiles que hayan ayudado a los miembros del equipo a ser más eficientes.
- Refactorización de Procesos: Similar a la refactorización de código, esta técnica implica revisar y ajustar los procesos actuales para hacerlos más eficientes, eliminando pasos innecesarios o redundantes.
- Reuniones de Alineación de Equipos (Team Alignment Meetings): Organizar reuniones breves y focalizadas donde los equipos puedan alinear expectativas, definir objetivos claros y discutir estrategias para alcanzarlos con éxito.
- Workshops de Resolución de Problemas: Organizar talleres donde los equipos puedan colaborar para encontrar soluciones a problemas específicos, utilizando técnicas como el brainstorming o el pensamiento de diseño (Design Thinking).

6. Consideraciones finales

A medida que llegamos al final de este recorrido, es imperativo que nos detengamos un momento para contemplar el viaje transformador que People Information LLC emprendió, un espejo del potencial que Scrum tiene para revolucionar las operaciones comerciales. Desde las recomendaciones pragmáticas para la adopción de Scrum hasta el desglose conciso de términos vitales, hemos cubierto terreno significativo, allanando el camino para futuros éxitos en el ámbito ágil. Puntos clave de este capítulo: - Recomendaciones para Adoptar Scrum: Estrategias fundamentadas para una transición exitosa a Scrum. - Glosario y Términos Usados: Clarificación de términos y conceptos centrales en Scrum. - Caso de Éxito de People Information LLC: Una mirada a los beneficios tangibles y la transformación alcanzada por People Information LLC con la adopción de Scrum.

Consideraciones finales sobre el uso y adopción de Scrum

Al llegar al final de nuestro trayecto a través del marco de trabajo Scrum, es vital detenerse a considerar las profundas implicaciones que tiene la adopción de Scrum en una organización. Es mucho más que simplemente asimilar nuevas técnicas; se trata de cultivar un cambio cultural que impregne de agilidad y eficiencia cada rincón de la empresa.

En este último capítulo, analizaremos aspectos críticos para tener en cuenta durante la adopción de Scrum, desde entender su verdadera esencia hasta cómo puede escalar eficazmente, sin olvidar la vitalidad de mantener un equipo motivado. Además, subrayaremos la necesidad de una optimización continua para asegurar una travesía fructífera a largo plazo en el mundo ágil.

Ahora, vamos a explorar estas consideraciones cruciales que pueden determinar el éxito de tu travesía con Scrum.

¿Scrum se implementa o se adopta?

El marco de trabajo Scrum consiste más en un cambio cultural en toda la organización que en simplemente la implementación de las herramientas o técnicas aquí descritas, así que es erróneo pensar en la “implementación” de Scrum, lo más apropiado es hablar de la “adopción” de Scrum, ya que se necesita poner en marcha las prácticas propuestas, realizar experimentos y estar ajustando continuamente el marco de trabajo a las necesidades cambiantes de la organización.

Para la adopción de las prácticas Scrum se requiere el apoyo de toda la organización, esta cultura orientada hacia lo ágil está basada en la simplicidad, el valor y el alto rendimiento.

¿Es valioso trabajar con procesos?

Si bien, en muchas organizaciones trabajar con procesos es visto como una gran ventaja porque asegura el buen desempeño de sus colaboradores, cuando hablamos de entornos ágiles, tendremos que ver la situación desde otro punto.

Las organizaciones cuentan con modelos de operación que tienden a comportarse de manera “estática”, en los que generalmente encontramos una estructura de jerarquías y burocracia, donde de manera rutinaria los colaboradores conocen los productos/servicios de la organización y saben cómo operar en el día a día para garantizar su adecuada entrega; sin embargo con los proyectos no siempre ocurre lo mismo debido a que los miembros del equipo de proyecto comúnmente se enfrentan a nuevos desafíos que los impulsa a cambiar constantemente y tomar una conducta autónoma en determinadas situaciones.

Por otro lado, vemos que las organizaciones fácilmente pueden percibir los beneficios de los procesos (facilitan las revisiones de calidad, la inducción del personal, las auditorías, etc) siempre y cuando estos procesos se lleven a la práctica y no permanezcan como simples documentos, ya que por sí solos no traen valor a la organización. Esta premisa aplica tal cual, a los ambientes ágiles, pues cuando hablamos de procesos debemos mantener un enfoque hacia la práctica (que sean interiorizados y ejecutados naturalmente por todos los miembros).

burocracia
Image not found or type unknown

> Es importante mantener un sano equilibrio entre la típica burocracia organizacional (que desfavorece la agilidad) y la anarquía (que disminuye la credibilidad en el equipo). Cuando se encuentra y mantiene el punto medio entre estas dos, el equipo se encuentra en un entorno de agilidad: sin burocracia absoluta y sin anarquía absoluta.

Escalabilidad de Scrum

Scrum puede ser utilizado en grandes proyectos sin importar la cantidad de personas o actividades a desarrollar, esto siempre y cuando se respete la regla de formar pequeños equipos de personas, ya que este tipo de equipos son muy flexibles y adaptivos. Las fortalezas de los equipos Scrum que permiten su fácil escalamiento son:

- Pueden operar individualmente, en varios, muchos, e incluso en redes de equipos.
- Su capacidad multifuncional les permite desarrollar, lanzar, operar y mantener el trabajo y el producto con la menor cantidad de dependencias posibles.
- La colaboración y la interacción continua entre equipos y con el cliente les permite desarrollar arquitecturas más sofisticadas en los entornos de desarrollo.

Ahora bien, existen múltiple publicaciones donde puedes encontrar más información sobre este tema, citamos algunas a continuación:

- **Large-Scale Scrum (LeSS)**: Una estructura de Scrum a gran escala que conserva muchos de los principios y prácticas de Scrum en un nivel más amplio.

- **SAFe (Scaled Agile Framework)**: Un marco que proporciona una guía para trabajar en escala grande, integrando múltiples equipos Scrum hacia un objetivo común.

Motivación del equipo

Cuando se trabaja en entornos de trabajo multidisciplinarios, la motivación es uno de los aspectos más importantes a tener en cuenta para conseguir un entorno de trabajo colaborativo y altamente competitivo, para lo cual es necesario promover a través del aprendizaje y la confianza, el desarrollo de habilidades de comunicación entre el colaborador y la empresa que garanticen el cumplimiento de las metas y retos individuales de cada una de las partes.

Es aquí donde queremos destacar el esfuerzo realizado por el autor Jurgen Appelo, en su publicación “Management 3.0”.

¡Hasta la Próxima Aventura Ágil!

Ha sido un viaje enriquecedor, explorando juntos las diversas dimensiones de Scrum. Esperamos que este manual no solo haya ampliado tu conocimiento, sino que también haya sembrado una semilla de curiosidad y pasión por continuar profundizando en el universo ágil.

Queremos expresar nuestro más sincero agradecimiento por permitirnos ser parte de tu travesía en el mundo de Scrum. No es una despedida, sino un hasta pronto, porque la agilidad es un viaje continuo de aprendizaje y crecimiento.

¡Gracias por confiar en CertMind para guiarte en esta fase de tu camino ágil! Esperamos reencontrarnos en futuras exploraciones, aventuras y descubrimientos.

¡Hasta la próxima!

Anexo A: Adopción de Scrum en People Information LLC

Caso de Éxito: Transformación Ágil en People Information LLC

En la etapa inicial, People Information LLC se encontraba en una encrucijada crítica, lidiando con demoras significativas en la entrega de productos y una comunicación fragmentada entre los equipos. Los proyectos a menudo se desviaban de su curso, acumulando costos adicionales y minando la moral del equipo.

En este escenario crítico, la empresa reconocía que necesitaba una reestructuración significativa de su enfoque de gestión de proyectos. Por lo tanto, decidieron embarcarse en una jornada de transformación ágil, seleccionando Scrum como el catalizador de este cambio. Con el desarrollo de su próximo producto, un sofisticado software HRM en el horizonte, era imperativo que encontraran un enfoque más efectivo y ágil para gestionar el desarrollo del mismo.

Aunque no estuvieron exentos de enfrentar desafíos significativos.

En el camino, se toparon con una resistencia cultural significativa, con algunos miembros del equipo resistiéndose a dejar los métodos de trabajo anteriores. Además, hubo una comprensión inicial inadecuada de Scrum, lo que llevó a algunos malentendidos y aplicaciones incorrectas de sus principios en las primeras etapas.

Fase 1: Sensibilización y Formación Los empleados fueron capacitados en los fundamentos de Scrum, aunque enfrentaron problemas de falta de compromiso de algunos líderes senior, lo que ralentizó el proceso inicial de formación y sensibilización.

Fase 2: Planificación y Mapeo Se estableció un Product Goal claro, pero surgieron desafíos en cuanto a la gestión del Product Backlog, con algunos ítems insuficientemente definidos o incorrectamente priorizados. Esto condujo a ciertos problemas en la fase de mapeo, con equipos luchando por definir un MVP coherente y alcanzable.

Fase 3: Implementación y Adaptación Se formaron pequeños equipos multifuncionales que comenzaron a trabajar en sprints de dos semanas. A pesar de los problemas iniciales de colaboración y algunos conflictos derivados de una cultura organizacional inicialmente rígida, los equipos comenzaron a encontrar su ritmo, aprendiendo a colaborar más efectivamente y a adaptarse con más flexibilidad a los cambios.

A lo largo de la adopción, People Information LLC trabajó arduamente para superar estos obstáculos, ajustando su enfoque y aprendiendo de los errores iniciales. En pocos meses, notaron una entrega de productos más rápida y eficiente, junto con una mejora significativa en la satisfacción del cliente y la colaboración entre equipos.

La adopción de Scrum no solo transformó su proceso de desarrollo de productos sino que también cultivó una cultura de colaboración y adaptabilidad en toda la organización, demostrando que, a pesar de los desafíos, la adopción de Scrum puede conducir a resultados positivos y significativos.

Algunos elementos clave que ayudaron a People Information LLC a alcanzar el éxito a pesar de los desafíos:

1. **Liderazgo Comprometido:** Los líderes de la organización estuvieron firmemente comprometidos con la adopción de Scrum, proporcionando el apoyo necesario para superar obstáculos.
2. **Formación Continua:** Implementaron programas de formación continua para asegurar que todos los miembros del equipo estuvieran bien versados en los principios y prácticas de agilidad (no solo recibieron formación en Scrum).
3. **Comunicación Mejorada:** A pesar de los problemas iniciales, trabajaron para mejorar la comunicación entre equipos y eliminar los silos de información.
4. **Ciclos Iterativos de Mejora:** Utilizaron los ciclos iterativos de Scrum para aprender y ajustar continuamente su enfoque, permitiéndoles mejorar con cada sprint.
5. **Feedback del Cliente Incorporado:** Aseguraron que el feedback del cliente estaba incorporado de manera regular en el proceso de desarrollo, lo que les ayudó a crear un producto más alineado con las necesidades del cliente.
6. **Cultura de Colaboración:** Fomentaron una cultura de colaboración, donde cada miembro del equipo estaba motivado para trabajar juntos hacia un objetivo común.
7. **Resiliencia y Adaptabilidad:** A pesar de los desafíos, mostraron una notable resiliencia y adaptabilidad, ajustándose a las lecciones aprendidas y perseverando a pesar de los obstáculos.

Estos elementos clave no solo ayudaron a People Information LLC a superar los desafíos de la adopción de Scrum, sino que también fueron fundamentales para su eventual éxito. Espero que estos puntos reflejen bien el espíritu de su transformación ágil.

Glosario y anglicismos

Anglicismos

En este libro se utilizan varios términos utilizados originalmente en inglés, esto con el fin de no causar confusiones con las distintas traducciones que a estos se les pueden dar. A continuación, se encuentra el listado de términos que NO fueron traducidos en esta publicación, es decir, que se mantienen en su forma original:

- **Burndown chart:** Cuadro de quemado hacia abajo o avance
- **Developer:** Desarrollador, quien ejecuta o desarrolla el trabajo
- **Framework:** Marco de trabajo
- **Product Backlog:** Lista de pendientes del producto
- **Product Goal:** Objetivo de producto
- **Product Owner:** Propietario (Dueño) del producto
- **Scrum Board:** Tablero Scrum
- **Scrum Master:** Maestro Scrum
- **Sprint:** Iteración
- **Sprint Backlog:** Lista de pendientes del Sprint
- **Sprint Goal:** Objetivo del Sprint