Módulo 2. Metodologías de trabajo en el desarrollo del producto digital

Unidad 2.1 Metodologías de trabajo para la creación de productos digitales

2.1.1 *Agile*: fundamentos y beneficios en el desarrollo de productos digitales

Introducción a la metodología *Agile*

Agile es un marco de desarrollo iterativo e incremental que prioriza la colaboración constante, la adaptabilidad y la entrega continua de valor. Surgió en 2001 con la publicación del Manifiesto *Agile*, creado por expertos en desarrollo de software que buscaban superar las limitaciones de los métodos tradicionales basados en el sistema *waterfall*, un enfoque secuencial y rígido en el que cada fase debía completarse por completo antes de avanzar a la siguiente, lo que restringía la capacidad de respuesta ante cambios o retroalimentación durante el proceso. Por el contrario, *Agile* fue diseñado para que, en entornos técnicos y de desarrollo, los equipos pudieran dividir los proyectos en iteraciones cortas y entregar funcionalidades de manera continua. Los desarrolladores trabajan de forma colaborativa con diseñadores, evaluadores y otros perfiles técnicos para implementar soluciones de manera rápida y eficiente, con capacidad de responder a cambios en tiempo real. De este modo, los productos digitales se van desarrollando de forma iterativa y mucho más ágil.

A partir de este planteamiento, *Agile* se sustenta en los siguientes principios clave:

1. **Interacción por encima de procesos y herramientas.** La comunicación fluida entre equipos prevalece sobre la burocracia.
2. ***Software* funcional sobre documentación extensa.** Prioriza productos funcionales en lugar de reportes y extensas documentaciones.
3. **Colaboración con el cliente.** Involucra a los usuarios y *stakeholders* en cada fase del proceso para asegurar el éxito de cada iteración.

4. **Adaptación al cambio.** La flexibilidad es clave para responder a nuevas necesidades y *Agile* –el nombre mismo lo ilustra– es ideal para eso.

En el ámbito de los productos digitales deportivos, *Agile* aporta las siguientes ventajas:

- **Adaptación dinámica.** Permite reaccionar a cambios en competiciones o calendario de eventos deportivos o al cambio en las necesidades tan común en el ámbito deportivo.
- **Entrega incremental.** Facilita la implementación de mejoras graduales en apps, productos y plataformas. No hace falta esperar al final del proceso para tener un producto funcional y en producción.
- **Enfoque en el usuario.** Fomenta la creación de experiencias basadas en datos de uso reales, poniendo al usuario siempre en el centro.

Para comprender cómo esto se aplica en la práctica, pensemos en el ejemplo de un club de fútbol que decide adoptar *Agile* para desarrollar una plataforma de *streaming*. Partiendo de un producto mínimo viable y trabajando en *sprints* semanales que recogieran el *feedback* de los usuarios para optimizar la experiencia de visualización en vivo, podría conseguirse un aumento de hasta un 25 % en la retención de aficionados durante la primera temporada.

A continuación, presentamos una tabla comparativa entre *Agile* y *waterfall* que refleja sus diferencias en planificación, gestión del cambio y entrega de valor:

Tabla 1. Comparativa entre *Agile* y *waterfall*

Característica	*Agile*	*Waterfall*
Flexibilidad	Alta	Baja
Interacciones	Constantes	Etapas definidas
Feedback	Continuo	Al final del proyecto
Adaptación al cambio	Fácil	Difícil

Fuente: elaboración propia

Antes de la llegada de *Agile*, el desarrollo de productos seguía el modelo *waterfall*, en el que cada fase debía completarse por completo antes de iniciar la siguiente, sin posibilidad de retrocesos ni ajustes. Como consecuencia, con frecuencia se obtenían soluciones que ya no respondían a las necesidades cambiantes de los usuarios. Con *Agile*, esa dinámica cambió por completo: el enfoque iterativo permite reducir riesgos, incrementar la eficiencia y mejorar notablemente la satisfacción del cliente.

2.1.2 *Scrum*: roles, ceremonias y artefactos

Scrum es uno de los marcos de trabajo más populares dentro de *Agile*. Estructura el trabajo en ciclos cortos denominados sprints —generalmente de dos semanas— y se fundamenta en la definición de roles específicos y ceremonias establecidas. Este enfoque permite entregar valor de forma continua mediante iteraciones frecuentes, maximizando la eficiencia y la retroalimentación.

Roles en *Scrum*

Scrum contempla roles específicos que garantizan la coordinación y el avance continuo del equipo en cada *sprint*.

1. ***Product owner* (PO).** Responsable de priorizar los requerimientos y gestionar el *backlog* del producto (listado de tareas o mejoras). El PO representa la visión del cliente y asegura que el equipo trabaje en las tareas de mayor impacto para los objetivos del negocio.

2. ***Scrum master.*** Facilita el proceso, eliminando obstáculos y asegurando que el equipo siga los principios de *Scrum*. Actúa como un facilitador y protector del equipo, evitando interrupciones externas.

3. **Equipo de desarrollo.** Compuesto por diseñadores, desarrolladores y *testers*, el equipo es responsable de entregar los incrementos y mejoras de producto. Este equipo multidisciplinar trabaja de manera autónoma y colaborativa para alcanzar los objetivos que se definan para cada *sprint*. Muchas veces se lo conoce como *squad*.

Ceremonias de *scrum*

Esta metodología de trabajo se sustenta en una serie de ceremonias y eventos planificados de forma secuencial que facilitan la consecución de los objetivos y que conviene respetar.

- ***Daily standup.*** Consiste en una reunión breve —de unos 15 minutos cada mañana— en la que el equipo repasa el progreso del *sprint*. Durante el encuentro, cada integrante explica qué hizo ayer, qué hará hoy y si existe algún obstáculo que esté retrasando su trabajo. De este modo, es posible detectar bloqueos de forma inmediata y reajustar prioridades antes de que afecten el ritmo general del proyecto.

- ***Sprint planning.*** Es una sesión de planificación de cada *sprint*, con una duración de una a dos horas, que sirve para definir las tareas y establecer el objetivo del ciclo. Durante este encuentro, el equipo divide las funcionalidades en tareas manejables que debe completar al término del sprint, lo que garantiza que todos comprendan con claridad los entregables previstos.

- ***Sprint review.*** Revisión del incremento desarrollado para recibir *feedback* de los stakeholders. Dura aproximadamente 1 hora y se realiza al final de cada sprint. Se presenta el trabajo completado y se discuten posibles mejoras futuras.
    - **Impacto:** Fomenta la mejora continua del producto y asegura que los desarrollos cumplen las expectativas.

- ***Sprint retrospective.*** Evaluación de lecciones aprendidas al finalizar el *sprint*, con una duración de 1 hora y organizada justo después de la revisión. El equipo identifica puntos de mejora y fortalezas para implementar en futuros sprints. Esta ceremonia mejora la eficiencia y cohesión del equipo en futuros *sprints*.

Esta estructura garantiza que cada fase del sprint esté organizada y comunicada de manera adecuada y favorece la entrega incremental de valor cada pocas semanas. Contar con un equipo cohesionado y con baja rotación resulta especialmente beneficioso, pues facilita la optimización y el perfeccionamiento continuo de los procesos.

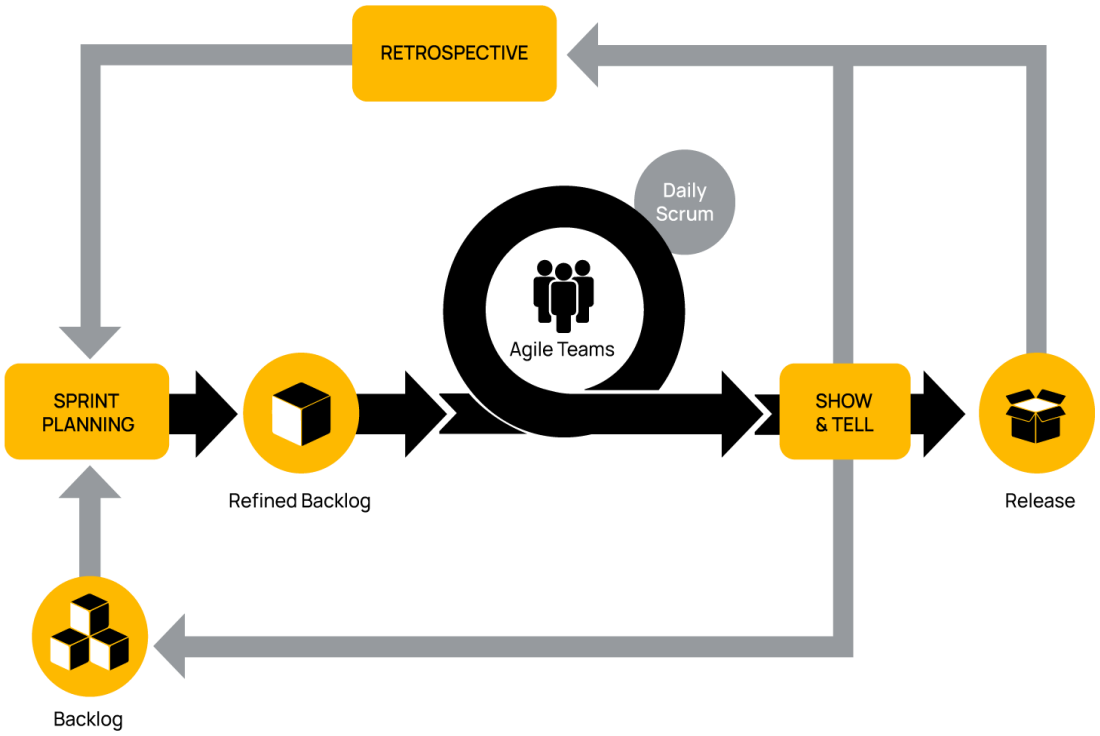
Artefactos de *Scrum*

En Scrum, los artefactos constituyen los elementos clave que permiten visibilizar el trabajo pendiente, planificado y finalizado, facilitando la transparencia y el seguimiento continuo del progreso del proyecto.

- ***Product backlog.*** Lista priorizada de funcionalidades y tareas. Se actualiza constantemente para reflejar las nuevas prioridades y necesidades que van surgiendo alrededor del producto. El *product owner* es el encargado de listar y ordenar por prioridad las mejoras que se quieren implementar en el producto.

- ***Sprint backlog.*** Conjunto de tareas seleccionadas para un *sprint*. Contiene las tareas detalladas necesarias para alcanzar el objetivo del *sprint*.
- ***Increment.*** Producto o funcionalidad lista para entrega al finalizar el *sprint*. Representa un avance tangible y funcional que se puede mostrar o usar.

Figura 1. Diagrama de flujos de trabajo en un marco *agile*



Fuente: elaboración propia

Las tareas y las nuevas necesidades deben estimarse de forma colaborativa para proyectar con mayor precisión el alcance del *sprint*, considerando tanto el tamaño del equipo como la complejidad de las tareas.

La cohesión y la práctica en un equipo *Agile* resultan fundamentales, ya que mejoran sus dinámicas y procesos de manera progresiva. A medida que los miembros trabajan juntos durante más tiempo, acumulan experiencia, profundizan en el conocimiento mutuo y del producto, lo que se traduce en una mayor eficiencia y en un incremento del valor entregado.

Ejemplo práctico

Durante el desarrollo de una *app* de *fan engagement* para la final de un torneo, el equipo organizó un *sprint* de dos semanas para incorporar notificaciones en tiempo real de goles y

estadísticas. En la planificación, el *product owner* priorizó esta funcionalidad por su impacto en la experiencia del usuario. Durante los *daily standups*, los desarrolladores compartieron su progreso y resolvieron bloqueos con la ayuda del *scrum master*. Al cierre del *sprint*, en la *sprint review*, los *stakeholders* probaron las notificaciones y propusieron ajustes en la personalización de los mensajes. Finalmente, en la retrospectiva, el equipo identificó la necesidad de mejorar la sincronización de datos y planificó optimizar las consultas al servidor en el siguiente sprint.

Implementación de *Scrum* en proyectos deportivos

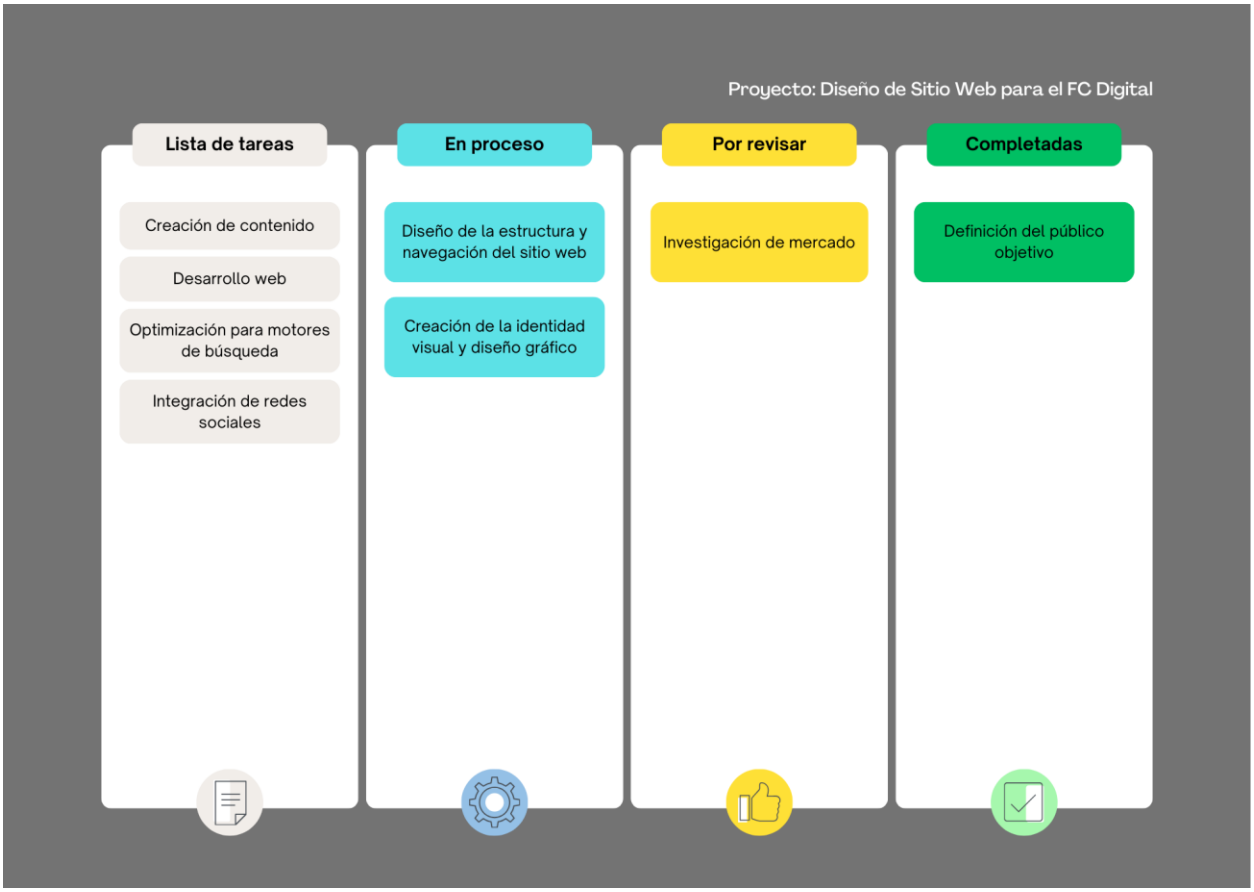
En proyectos digitales deportivos, *Scrum* posibilita una adaptación ágil a variaciones como cambios en la programación de partidos, nuevas peticiones de contenido o imprevistos. Por ejemplo, una plataforma OTT que emite eventos en directo puede emplear *Scrum* para planificar la mejora de la calidad de vídeo y llevar a cabo pruebas A/B de funcionalidades de interacción durante los encuentros.

2.1.3 *Kanban*: flujo visual y priorización continua

Principios de Kanban

Dentro de las metodologías de trabajo *agile*, Kanban es una opción visual que permite gestionar el flujo de trabajo a lo largo del proceso de desarrollo mediante un tablero dividido en columnas que representan el estado de las tareas (por ejemplo, «Por hacer», «En proceso» y «Completado»), tal como se puede observar en la siguiente figura.

Figura 2. Ejemplo de tablero Kanban para la gestión de tareas



Fuente: elaboración propia.

Kanban se puede utilizar para gestionar tareas de *software*, pero también para gestionar tareas del día a día o incluso proyectos. Es decir, la filosofía se puede aplicar a muchos casos de uso. Además, el tablero puede ser físico o virtual.

A continuación, se describen los elementos clave de Kanban:

- **Visualización del flujo de trabajo.** Permite identificar bloqueos y cuellos de botella. Por ejemplo, en un equipo de desarrollo de productos deportivos, durante un partido en vivo, se pueden visualizar tareas como actualizaciones de resultados y carga de contenido multimedia en tiempo real.
- **Limitación del WIP (*work in progress*).** Controla la cantidad de tareas simultáneas que se están llevando a cabo para evitar la saturación. Esto ayuda a priorizar, por ejemplo, actualizaciones críticas sobre contenido promocional durante momentos clave del partido.

- **Mejora continua.** Se basa en la retroalimentación constante para optimizar los flujos. Los equipos pueden realizar revisiones posteriores al evento para identificar puntos de mejora en la coordinación de las actualizaciones en vivo.

Para mostrar cómo *Kanban* y *Scrum* abordan la gestión del trabajo desde perspectivas distintas, a continuación se comparan sus principales características en aspectos fundamentales:

Tabla 2. Diferencias entre *Kanban* y *Scrum*

Característica	*Kanban*	*Scrum*
Estructura de trabajo	Flujo continuo	*Sprints*
Roles definidos	No obligatorios	Definidos (PO, *scrum master*)
Cambios durante el proceso	Permitidos	Restringidos a revisiones

Fuente: elaboración propia.

Para comprender la aplicación práctica de *Kanban*, imaginemos que el equipo de una *app* de *fan engagement* utiliza un tablero con las columnas «Planificación», «Producción», «Revisión» y «Publicado» para gestionar la publicación de contenidos durante la temporada. Al limitar el número de tareas en la columna «Revisión», garantizan que las entregas críticas, como las estadísticas de partidos en tiempo real, no se retrasen. Una vez concluida la temporada, analizan el flujo de trabajo para detectar áreas de mejora y ajustar los límites de WIP.

2.1.4. *Design Thinking* como complemento a *Agile*

Como ya hemos visto, *Design Thinking* es un método de identificación de mejoras centrado en el usuario que busca resolver problemas complejos mediante la empatía y la experimentación. Este enfoque se complementa de manera natural con *Agile*, pues aporta una base sólida para comprender las necesidades de los usuarios antes de comenzar a desarrollar e iterar soluciones. Además, *Design Thinking* enriquece la planificación de *sprint* y la retrospectiva al generar ideas orientadas a optimizar la iteración continua.

Ahora bien, **¿cómo integrar *Design Thinking* en *Agile*?**

Para integrar *Design Thinking* en *Agile*, el equipo inicia con la fase de descubrimiento, en la que emplea técnicas de empatía y prototipado —entrevistas, mapas de empatía y pruebas tempranas— para recopilar información de los usuarios y definir las funcionalidades clave. A continuación, las ideas validadas durante el prototipado se incorporan directamente en los *sprints*, donde se desarrollan, prueban y refinan mediante ciclos cortos de trabajo y *feedback* constante, garantizando así que las soluciones respondan profundamente a las necesidades reales antes de pasar a la siguiente iteración.

Para entender mejor esta integración, la presentaremos en etapas:

1. **Empatizar**. Antes de un nuevo *sprint*, se organiza un taller de descubrimiento con entrevistas a usuarios y análisis de datos para comprender mejor las expectativas del cliente.
2. **Definir**. Durante la planificación de *sprint*, el equipo convierte los hallazgos en historias de usuario claras y los integra en los objetivos de *sprint*.
3. **Idear**. Se llevan a cabo sesiones de ideación rápidas durante los refinamientos de *backlog* para explorar múltiples enfoques.
4. **Prototipar**. Dentro del *sprint*, se crean prototipos funcionales rápidos que se presentan en la revisión.
5. **Probar**. En la retrospectiva, se revisa el *feedback* obtenido tras pruebas de usuarios y se identifica qué iteraciones deben priorizarse en futuros *sprints*.

A modo de ejemplo, podemos pensar en el rediseño de una *app* de venta de entradas. El equipo utilizó *Design Thinking* para abordar los puntos de fricción en la experiencia de compra. Durante un taller inicial, identificaron que los usuarios encontraban confuso el proceso de selección de asientos. A partir de esta información, se implementaron las siguientes acciones:

- Se desarrolló un prototipo funcional en Figma que permitió a los usuarios seleccionar asientos de manera visual e intuitiva.
- Este prototipo se integró en un *sprint* ágil y se probó en un entorno controlado.
- Los resultados mostraron un aumento del 40 % en el tiempo promedio dedicado a la selección de asientos, lo que indicaba una experiencia más satisfactoria.

Con esto, el impacto se pudo medir y se obtuvieron los siguientes resultados:

- Reducción del 25 % en las tasas de abandono durante el proceso de compra.

- Incremento del 15 % en conversiones de ventas.
- Mejoras en la percepción de la marca gracias a la experiencia optimizada.

Conclusión

La combinación de metodologías como *Agile*, *Scrum*, *Kanban* y *Design Thinking* permite crear productos digitales sólidos y adaptables que facilitarán el éxito en un entorno tan cambiante como el deportivo. Estas metodologías promueven la colaboración, la entrega continua de valor y la personalización de la experiencia del usuario, garantizando un enfoque centrado en sus necesidades y en los objetivos de negocio.

Unidad 2.2 Herramientas y procesos colaborativos

2.2.1. Herramientas clave para la gestión de metodologías ágiles

El uso de herramientas colaborativas es esencial para gestionar los flujos de trabajo en un entorno ágil. Estas herramientas permiten planificar tareas, asignar responsables y realizar un seguimiento continuo del progreso del proyecto.

En proyectos de productos digitales deportivos, las herramientas facilitan la coordinación en tiempo real incluso durante partidos y eventos clave, asegurando que los equipos puedan reaccionar lo más rápidamente posible.

Principales herramientas para la gestión ágil

A continuación, presentamos un listado de las herramientas utilizadas actualmente para la gestión ágil de proyectos.

1. **Jira**

Jira es una herramienta especializada en la gestión de proyectos *agile* y *sprints*, que permite crear historias de usuario, organizar tareas y generar informes detallados sobre el avance del equipo. Además, cuenta con numerosos conectores que facilitan la integración con plataformas de correo, mensajería, diseño y documentación, entre otras. En un proyecto deportivo, por ejemplo, se puede emplear Jira para definir historias de usuario que describan funcionalidades como la personalización de notificaciones en una *app* de *fan engagement*, asignar las tareas correspondientes a cada integrante, hacer seguimiento de bloqueos y medir el progreso de cada *sprint* mediante tableros *Kanban* o gráficos de *burndown*. De este modo, Jira contribuye a mantener la visibilidad y la coordinación necesarias para entregar valor de forma continua y adaptarse a las necesidades cambiantes de los aficionados.

2. ***Trello***

Trello se basa en tableros *Kanban* que facilitan la gestión visual de tareas, permitiendo dividir el flujo de trabajo en columnas como «Preproducción», «Revisión» y «Publicación». En el contexto de una plataforma OTT durante eventos en vivo, Trello puede emplearse para coordinar la producción de contenido: desde la planificación previa hasta la aprobación y el envío final de vídeos o gráficos en directo, garantizando que cada etapa avance con claridad y sincronía entre los equipos de edición, streaming y redes sociales.

3. **Monday**

Monday ofrece paneles personalizables y herramientas para priorizar tareas, lo que facilita el seguimiento de recursos y la asignación de prioridades en proyectos diversos. Es especialmente recomendable para coordinar recursos en iniciativas que involucran varias áreas, como diseño, desarrollo y *marketing*.

Asimismo, podemos mencionar una serie de herramientas de colaboración en tiempo real que sirven para la gestión de proyectos ágiles.

1. **Slack.** Facilita la comunicación instantánea e integra notificaciones procedentes de herramientas como Jira y Trello. Un ejemplo práctico consiste en crear canales específicos para coordinar actualizaciones en tiempo real durante un torneo.

2. **Figma**. Esta plataforma colaborativa permite diseñar interfaces y prototipos, especialmente indicada para elaborar prototipos interactivos de aplicaciones deportivas que reciban retroalimentación directa de los *stakeholders*.

3. **Miro.** Esta herramienta facilita la elaboración de mapas de ideas y la planificación visual; por ejemplo, durante reuniones de estrategia puede emplearse para diseñar roadmaps y flujos de trabajo.

Ahora que sabemos qué herramientas son útiles, cabe preguntarnos: **¿bajo qué criterios se selecciona una herramienta u otra?** Esto dependerá de varios factores, a saber:

- **Tamaño del equipo.** Equipos pequeños pueden beneficiarse más de herramientas simples como Trello, mientras que equipos grandes requieren plataformas robustas como Jira.

- **Tipo de proyecto.** Proyectos de diseño pueden priorizar Figma, mientras que los centrados en desarrollo técnico se benefician más de Jira.

- **Integraciones.** Asegurar que las herramientas seleccionadas puedan conectarse con otras ya utilizadas por el equipo.

Ejemplo práctico

Un equipo de desarrollo de productos digitales deportivos implementó un flujo ágil para optimizar la funcionalidad de su *app* de *streaming* en vivo. Durante el *sprint*, utilizaron Trello para gestionar las tareas relacionadas con pruebas A/B de nuevas interfaces y optimización de *buffer* en diferentes dispositivos. Con reuniones diarias de sincronización en Slack, identificaron rápidamente errores y ajustaron el enfoque según los datos en tiempo real. Este

proceso redujo el tiempo de carga de videos en un 35 % y mejoró la experiencia del usuario durante eventos masivos.

2.2.2 Procesos de colaboración en equipos multidisciplinares

En los proyectos ágiles, los equipos multidisciplinares reúnen con frecuencia perfiles que abarcan desde el diseño y el desarrollo hasta el negocio y el análisis de datos. Esta diversidad de competencias potencia los resultados, aunque también plantea retos en la coordinación y la comunicación. En el entorno digital deportivo, donde se trabaja bajo presión para ofrecer resultados en tiempo real, una colaboración eficaz resulta fundamental para evitar cuellos de botella y garantizar entregas exitosas.

A continuación, presentamos prácticas orientadas a potenciar la comunicación efectiva y a facilitar la gestión de conflictos en equipos ágiles.

1. **Prácticas para la comunicación efectiva**

- **Establecimiento de roles y responsabilidades claras.** El establecimiento de roles y responsabilidades claras evita solapamientos y duplicación de tareas: al documentar las funciones de cada miembro, un diseñador UX puede liderar la experiencia del usuario, mientras que un analista de datos aporta insights para personalizar la *app* de *fan engagement*.

- **Reuniones breves y efectivas**. Las reuniones breves y efectivas limitan las *dailies* a 15 minutos diarios, centrándose en lo realizado, lo que se hará y los bloqueos, lo cual mejora el enfoque del equipo y minimiza las interrupciones.

- **Documentación visual.** La documentación visual, a través de diagramas de flujo y tableros *Kanban*, alinea al equipo en torno a las prioridades y avances, favoreciendo un avance sólido hacia el objetivo común.

2. **Gestión de conflictos y sincronización**

La gestión de conflictos y la sincronización en equipos ágiles requiere reconocer que con frecuencia surgen diferencias en prioridades entre áreas como diseño y desarrollo, así como problemas de coordinación en equipos distribuidos; para abordarlos, conviene facilitar sesiones de retroalimentación periódicas que alineen expectativas y emplear herramientas colaborativas como Confluence o Notion para centralizar la documentación y garantizar la transparencia necesaria para mantener a todos sincronizados.

3. **Herramientas de apoyo**

- **Slack**. Facilita la creación de canales dedicados a equipos o proyectos, lo que mejora la comunicación, ahorra tiempo y evita reuniones improductivas; por ejemplo, se puede habilitar un canal exclusivo para gestionar actualizaciones durante un torneo importante.
- **FigJam y Miro.** Resultan ideales para sesiones de «brainstorming» y la planificación visual de flujos de usuarios, y su uso está especialmente recomendado para diseñar roadmaps colaborativos en proyectos a largo plazo.
- **Google Docs y Notion.** Permiten compartir documentación actualizada y garantizar que todos los miembros tengan acceso a la información más reciente.

Ejemplo práctico

En el desarrollo de una OTT deportiva, el equipo de diseño preparó *wireframes* en Miro, mientras los desarrolladores crearon prototipos interactivos en Figma. Los analistas de datos aportaron métricas sobre la audiencia para priorizar funcionalidades clave. Gracias a reuniones diarias y una sesión de retroalimentación semanal, el equipo cumplió con los plazos de lanzamiento y entregó una experiencia optimizada para los usuarios.

Conclusión

Los procesos de colaboración en equipos multidisciplinarios son esenciales para proyectos ágiles exitosos. Al establecer roles claros, fomentar una comunicación efectiva y utilizar herramientas colaborativas, los equipos pueden superar los desafíos y lograr entregas de alto impacto, especialmente en entornos deportivos dinámicos donde los plazos y las expectativas son críticos.

2.2.3 Optimización de flujos de trabajo

Identificación de cuellos de botella

Los retrasos en las entregas son uno de los obstáculos más frecuentes en el desarrollo de productos. A menudo surgen inconvenientes, bloqueos o circunstancias ajenas al equipo que afectan la planificación. Cuando no se deben a imprevistos o causas de fuerza mayor, estos retrasos suelen originarse en la falta de comunicación, la acumulación de tareas o procesos ineficientes. Por ello, resulta fundamental gestionar de forma óptima los flujos de trabajo e implantar herramientas visuales y metodologías que permitan identificar áreas problemáticas y maximizar la eficiencia.

Ahora bien, hay ciertos indicadores de cuellos de botella que permiten que los equipos se adelanten a buscar la solución.

- **Tareas prolongadas en «En Progreso»:** las tareas que permanecen demasiado tiempo en esta fase del tablero indican bloqueos o falta de recursos.
- **Incremento de solicitudes de soporte:** si el equipo recibe constantes aclaraciones o cambios, puede ser señal de una definición inicial de requisitos deficiente.
- **Reuniones de emergencia frecuentes:** cuando los problemas no se resuelven de manera proactiva, aumentan las reuniones no planificadas.

Todos estos factores pueden afectar la planificación, consumir recursos y tiempos no previstos y, en consecuencia, reducir la calidad y el valor entregado, además de prolongar los plazos de entrega.

Estrategias para la optimización

Para evitar estas circunstancias y favorecer la buena marcha del equipo, hay una serie de estrategias que pueden resultar útiles a la hora de desarrollar productos.

1. **Automatización de procesos.** Implementar *scripts* y herramientas que reduzcan tareas repetitivas. Por ejemplo, automatizar pruebas unitarias o actualizaciones de bases de datos en apps deportivas.
2. **División de tareas complejas.** Desglosar tareas grandes en subtareas más pequeñas y manejables. Esto permite a los equipos avanzar en paralelo y mantener un ritmo constante.
3. **Priorización clara mediante *frameworks*.** Utilizar métodos como MoSCoW (Must, Should, Could, Won't) para garantizar que las tareas críticas se realicen primero. Estos métodos se tratan en otros apartados del curso.
4. **Monitorización del flujo de trabajo.** Tableros *Kanban* o herramientas como Trello y Jira permiten visualizar el estado actual de cada tarea y detectar bloqueos rápidamente.

Además de las estrategias mencionadas, la clave está en implementar un ciclo de retroalimentación y mejora continua. Esto se traduce en las siguientes acciones:

- Implementar revisiones periódicas de los flujos de trabajo mediante retrospectivas quincenales es una buena práctica que permite refinar los flujos de trabajo.

- Usar métricas como tiempo de ciclo y tiempo de entrega para evaluar la eficiencia del equipo y la capacidad de entregar valor.

Ejemplo práctico

Un equipo de desarrollo de *app* móvil para una liga deportiva enfrentaba retrasos constantes debido a bloqueos en la revisión de código. Decidieron automatizar las revisiones iniciales mediante *pull requests* automáticos y programaron reuniones semanales para resolver dudas técnicas. Este cambio redujo el tiempo total de despliegue en un 30 % y mejoró la moral del equipo al reducir frustraciones.

Conclusión

La optimización de flujos de trabajo —o *ways of working*— no solo incrementa la eficiencia operativa, sino que también promueve un entorno más colaborativo y orientado a resultados. Identificar cuellos de botella, priorizar tareas y aprovechar herramientas de automatización son pasos fundamentales para que los equipos se adapten con rapidez a las exigencias de un entorno dinámico como el deportivo digital, favoreciendo la entrega de valor, la calidad del trabajo y la moral de las *squads*.

2.2.4 Escalabilidad en equipos y procesos

A medida que los productos digitales crecen en alcance y complejidad, escalar los equipos y hacer que los procesos de trabajo no se resientan se convierte en una necesidad crítica. La escalabilidad no solo implica aumentar el tamaño del equipo, sino también garantizar que las operaciones sigan siendo ágiles y eficientes en contextos más grandes y dinámicos. Esto es particularmente relevante en proyectos deportivos digitales, donde la demanda puede variar drásticamente según eventos y temporadas.

A continuación, presentamos un listado con buenas prácticas que permiten escalar los procesos:

1. **Definición de subequipos o *squads* especializados**

Definir subequipos especializados—o *squads*—permite dividir al equipo general en grupos centrados en áreas como diseño UX, *backend*, *frontend* o análisis de datos, de modo que cada integrante desarrolle una experiencia profunda sin perder de vista los objetivos comunes. Por ejemplo, en el desarrollo de una *app* deportiva podría formarse un subequipo dedicado a la personalización de notificaciones y otro encargado de optimizar el rendimiento en dispositivos móviles, garantizando así agilidad y calidad en cada área.

2. **Coordinación mediante *Scrum* de *Scrums*:**

La coordinación mediante *Scrum de Scrums* consiste en celebrar reuniones semanales de sincronización en las que los líderes de cada subequipo comparten avances, retos y necesidades de apoyo, garantizando así la alineación entre los equipos y evitando esfuerzos duplicados o tareas redundantes.

3. **Uso de tableros compartidos**

La utilización de tableros compartidos permite consolidar la información clave en paneles visuales globales —por ejemplo, en Jira o Monday.com— que muestran el progreso general del proyecto y las interdependencias entre los *squads*, facilitando así la planificación y el seguimiento de las prioridades a nivel global.

4. **Estandarización de procesos**

La estandarización de procesos implica establecer un conjunto de reglas y guías comunes que garantizan la coherencia en la forma en que los equipos documentan, desarrollan y entregan sus tareas, un enfoque especialmente valioso en entornos con equipos numerosos o distribuidos.

Desafíos comunes y soluciones

A continuación repasaremos las situaciones y retos más comunes en este campo y aportaremos posibles soluciones.

- **Descoordinación entre equipos**

Solución: usar herramientas como Confluence para centralizar la documentación y realizar reuniones regulares de alineación.

- **Duplicidad de tareas**

Solución: implementar herramientas de integración que sincronicen el trabajo entre plataformas y definan responsables claros para cada tarea.

- **Pérdida de agilidad**

Solución: limitar el número de participantes en reuniones generales y promover la toma de decisiones a nivel de subequipo, escalando solo los bloqueos críticos.

Herramientas para facilitar la escalabilidad

Si bien ya las presentamos anteriormente, es útil resaltar que estas herramientas también son útiles para escalar procesos:

- **Slack.** Canales dedicados para la comunicación entre subequipos.
- **Jira y Trello.** Gestión de proyectos a nivel macro y micro.
- **Notion.** Centralización de documentación y procesos estándar.
- **Miro.** Coordinación visual para roadmaps y estrategias.

Ejemplo práctico

Durante el rediseño de una plataforma de *streaming* para una liga deportiva, el equipo inicial de 10 personas creció a 30 para gestionar la carga de trabajo adicional. Se crearon tres subequipos:

1. **Interfaz móvil.** Encargado de rediseñar la experiencia del usuario en dispositivos móviles.
2. **Optimización OTT.** Focalizado en mejorar el rendimiento del *streaming* y la calidad del *player* y del video.
3. **Funcionalidades *premium*.** Responsable de integrar herramientas de personalización y segmentación de usuarios.

Se realizaron reuniones semanales de *Scrum* de *Scrums* para compartir avances y coordinar dependencias. Además, se utilizó Jira para consolidar todas las tareas en un tablero maestro. Gracias a estas prácticas, el producto fue lanzado sin demoras, con un aumento del 25 % en la retención de usuarios en su primera semana.

Conclusión

Escalar equipos y procesos de manera efectiva requiere planificación, herramientas adecuadas y una estructura organizativa que permita la autonomía sin perder la coherencia. Al dividir responsabilidades, estandarizar procesos y mantener una comunicación clara, los equipos pueden enfrentar los desafíos de los proyectos complejos sin comprometer la calidad ni la velocidad de entrega. Esto es clave en entornos deportivos digitales, donde la demanda de los usuarios y los *stakeholders* puede cambiar rápidamente.

Referencias bibliográficas de consulta

Brown, T. (2009). *Change by design: How design thinking transforms organizations and inspires innovation*. Harper Business.

Cagan, M. (2017). *Inspired: How to create tech products customers love*. Wiley.

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum—Making the most of both*. Lulu Press, Inc.

Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Sutherland, J. (2014). *Scrum: The art of doing twice the work in half the time*. Crown Business.