SERVICIO NACIONAL DE APRENDIZAJE (SENA)

CENTRO DE INNOVACIÓN Y DESARROLLO TECNOLÓGICO Y DE LA INDUSTRIA DEL MUEBLE (CDITI)

Número de ficha: 3172293

Integrantes:

Fabian Guzman

Maycol Silva

James Machado

Frayin Parra

1. Introducción a las Metodologías Ágiles

Las **Metodologías Ágiles** son un conjunto de enfoques de gestión de proyectos que buscan una mayor flexibilidad, adaptabilidad y colaboración. A diferencia de los métodos tradicionales (como el "modelo en cascada"), las metodologías ágiles no planifican todo el proyecto desde el inicio, sino que lo dividen en ciclos cortos e iterativos. Esto permite que el equipo de trabajo y el cliente colaboren de manera constante, adaptando el proyecto a los cambios y entregando valor de forma continua.

2. Metodologías Ágiles más destacadas

Existen diversas metodologías ágiles, cada una con sus propias características y enfoques. Las más relevantes son:

Scrum

- **Descripción:** Scrum es la metodología ágil más popular y una de las más utilizadas en la gestión de proyectos, especialmente en el desarrollo de software. Es un marco de trabajo que se basa en la entrega de productos funcionales en ciclos cortos y fijos, llamados "sprints". Su objetivo principal es permitir a los equipos trabajar de forma más eficiente y adaptativa, respondiendo a los cambios del cliente y del mercado. Se enfoca en la transparencia, la inspección y la adaptación.

- **Principios y valores clave:**
    - **Empirismo:** Scrum se basa en el conocimiento que surge de la experiencia y en la toma de decisiones basada en lo observado.
    - **Roles:** Se definen tres roles específicos para asegurar la colaboración y la claridad en el proceso:
        - **Product Owner:** Es el responsable de maximizar el valor del producto. Mantiene el *Product Backlog* (una lista priorizada de todo lo que se necesita en el producto) y representa la voz del cliente.
        - **Scrum Master:** Actúa como un facilitador y "coach" del equipo. Su rol es asegurar que el equipo siga los principios de Scrum, eliminando obstáculos y protegiendo al equipo de interrupciones externas.
        - **Equipo de Desarrollo:** Son las personas encargadas de crear el producto. Son auto-organizados y multifuncionales, lo que significa que tienen todas las habilidades necesarias para completar el trabajo.

- **Eventos:** Scrum define una serie de eventos fijos que crean un ritmo constante en el proyecto:
  - **Sprint Planning:** Al inicio de cada sprint, el equipo planifica qué trabajo se realizará.
  - **Daily Scrum:** Una reunión diaria de 15 minutos para que el equipo sincronice sus actividades y revise el progreso hacia la meta del sprint.
  - **Sprint Review:** Al final del sprint, el equipo presenta el trabajo completado al Product Owner y a otros interesados.
  - **Sprint Retrospective:** El equipo reflexiona sobre el sprint que acaba de terminar e identifica mejoras para el próximo.

Ejemplo práctico de Scrum:

Imagina un equipo que está creando una aplicación móvil. Deciden trabajar con Scrum en sprints de dos semanas.

- **Antes del sprint:** El Product Owner tiene una lista de funcionalidades (*Product Backlog*).
- **Sprint Planning (Lunes de la semana 1):** El equipo se reúne y decide que en las próximas dos semanas trabajarán en la función de "perfil de usuario". Elige las tareas del *Product Backlog* para completar esa función.
- **Daily Scrum (Cada mañana):** Durante 15 minutos, el equipo comparte: ¿Qué hiciste ayer?, ¿qué harás hoy?, ¿hay algún obstáculo?
- **Sprint Review (Viernes de la semana 2):** El equipo muestra al Product Owner y a otros interesados la función de "perfil de usuario" ya funcionando. Se discute qué se logró y si hay ajustes que hacer.
- **Sprint Retrospective (Viernes de la semana 2, después del *review*):** El equipo reflexiona sobre el sprint: "Hemos tenido problemas con la comunicación, deberíamos usar una herramienta de mensajería más eficiente". Esta mejora se aplicará en el próximo sprint.

Kanban

- **Descripción:** Kanban es una metodología ágil que se enfoca en la visualización del flujo de trabajo y la gestión de tareas. A diferencia de Scrum, que se basa en ciclos de tiempo fijos (sprints), Kanban es un método de flujo continuo. Su objetivo principal es maximizar la eficiencia, reducir los cuellos de botella y entregar el trabajo tan pronto como esté listo.

- **Principios clave:**
    - **Visualizar el flujo de trabajo:** La herramienta principal de Kanban es un tablero visual (físico o digital) con columnas que representan las diferentes etapas del proceso (por ejemplo: "Pendiente", "En progreso", "Finalizado").
    - **Limitar el trabajo en curso (WIP - Work In Progress):** Se establece un límite en la cantidad de tareas que pueden estar en la columna "En progreso". Esto evita que el equipo se sature y se concentre en terminar las tareas antes de empezar otras nuevas.
    - **Gestionar el flujo:** El equipo se enfoca en que las tareas se muevan de forma fluida a lo largo del tablero, identificando y eliminando cualquier interrupción que ralentice el proceso.
    - **Hacer las políticas explícitas:** Las reglas del proceso (qué significa que una tarea esté "lista para pasar" a la siguiente columna) deben estar claras y ser conocidas por todo el equipo.
    - **Mejorar de forma colaborativa:** El equipo reflexiona constantemente sobre cómo puede mejorar su flujo de trabajo y la eficiencia.

Ejemplo práctico de Kanban:

Imagina un equipo de marketing digital que gestiona la creación de contenido. Deciden usar un tablero Kanban con las siguientes columnas:

- **Ideas:** Aquí se colocan todas las ideas de contenido (artículos de blog, videos, publicaciones en redes sociales) que necesitan ser producidas.

- **En redacción:** Cuando un escritor empieza a trabajar en una idea, la mueve a esta columna. El equipo puede establecer un límite de WIP de 3 para asegurar que no haya demasiados borradores a la vez.
- **En revisión:** Una vez que el borrador está listo, pasa a esta columna para que otro miembro del equipo lo revise.
- **Aprobado:** El contenido revisado y finalizado se mueve aquí, listo para ser publicado.
- **Publicado:** Cuando el contenido está en línea, se mueve a esta columna para indicar que el trabajo está completamente terminado.

Extreme Programming (XP)

- **Descripción:** Extreme Programming (XP) es una metodología ágil que se enfoca en el desarrollo de software de alta calidad. Fue creada para mejorar la productividad y la capacidad de los equipos de desarrollo para responder a los requisitos cambiantes de los clientes. XP no es solo un conjunto de prácticas, sino una filosofía que promueve la comunicación, la simplicidad, la retroalimentación y el coraje.

- **Principios y prácticas clave:** XP se basa en un conjunto de prácticas técnicas y de equipo que lo diferencian de otras metodologías.
    - **Programación en parejas:** Dos desarrolladores trabajan juntos en un mismo ordenador. Mientras uno escribe el código, el otro lo revisa y busca fallos, lo que mejora la calidad y reduce los errores.
    - **Desarrollo guiado por pruebas (TDD - Test-Driven Development):** Se escriben pruebas automatizadas *antes* de escribir el código. El desarrollador solo escribe el código suficiente para que la prueba pase.
    - **Integración continua:** El código se integra y prueba varias veces al día. Esto evita grandes problemas de integración al final del proyecto y mantiene el sistema siempre funcional.
    - **Refactorización:** El equipo mejora constantemente la estructura del código sin cambiar su funcionalidad. Esto mantiene el código limpio, simple y fácil de mantener a largo plazo.
    - **Cliente en el sitio:** El cliente o un representante trabaja directamente con el equipo de desarrollo, lo que asegura que el producto se construya exactamente como se necesita.

    - **Historias de usuario:** Los requisitos del proyecto se escriben desde la perspectiva del cliente, en frases sencillas y no técnicas.

Ejemplo práctico de XP:

Imagina un equipo que está construyendo un nuevo sistema de reservas para un hotel. Con XP, el proceso sería así:

- **Historias de usuario:** El equipo se reúne con el gerente del hotel para escribir historias como: "Como cliente, quiero ver las habitaciones disponibles para una fecha específica".
- **TDD:** Antes de escribir el código para buscar habitaciones, un desarrollador escribe una prueba que verifique si el sistema devuelve las habitaciones correctas para esa fecha.
- **Programación en parejas:** Dos programadores trabajan juntos para escribir el código que haga que esa prueba pase. Uno escribe el código mientras el otro revisa y mejora.
- **Integración continua:** A lo largo del día, el nuevo código se sube al repositorio central, se combina con el trabajo de otros y se ejecuta automáticamente una suite de pruebas para asegurar que nada se rompa.
- **Refactorización:** Durante el proceso, el equipo nota que hay una parte del código que es confusa. Dedican tiempo a limpiarla y organizarla, asegurando que sea fácil de entender para todos.

Lean Software Development

- **Descripción:** Lean Software Development, o desarrollo de software "esbelto", es una metodología ágil que se basa en los principios de la manufactura *Lean* de Toyota. Su objetivo principal es maximizar el valor entregado al cliente mientras se minimizan los "desperdicios", es decir, cualquier actividad o recurso que no aporta valor al producto final. Se enfoca en la eficiencia, la rapidez y el aprendizaje continuo.

- **Principios clave:** Lean Software Development se guía por siete principios que buscan optimizar el proceso de desarrollo de principio a fin:
    1. **Eliminar el desperdicio:** Identificar y eliminar cualquier cosa que no aporte valor, como la documentación excesiva, reuniones innecesarias, código sin usar o esperas prolongadas.
    2. **Amplificar el aprendizaje:** Fomentar el conocimiento a través de la retroalimentación rápida y frecuente. El equipo debe aprender constantemente de los errores y del cliente.
    3. **Decidir lo más tarde posible:** Retrasar las decisiones importantes hasta tener la mayor cantidad de información posible. Esto permite mayor flexibilidad y adaptabilidad.
    4. **Entregar lo más rápido posible:** Entregar software funcional en ciclos cortos para obtener retroalimentación del cliente y validar las ideas rápidamente.
    5. **Potenciar al equipo:** El equipo de desarrollo tiene la autoridad y la responsabilidad de tomar decisiones, lo que fomenta la motivación y la creatividad.
    6. **Construir con calidad:** La calidad debe ser una prioridad desde el inicio. El objetivo es construir un producto robusto que no necesite un retrabajo constante.
    7. **Ver el todo:** Analizar el proyecto como un sistema completo, no como partes aisladas. El equipo debe entender cómo cada pieza encaja para ofrecer valor.

Ejemplo práctico de Lean Software Development:

Imagina un equipo que está creando un nuevo servicio de suscripción. Con un enfoque Lean, el equipo podría:

- **Eliminar desperdicio:** Deciden no crear un documento de requisitos de 100 páginas. En su lugar, utilizan un boceto de las funcionalidades principales y validan la idea con el cliente a través de una llamada corta.
- **Entregar rápido:** En lugar de construir todo el servicio, lanzan una versión muy básica con la funcionalidad principal de suscripción. Esto les permite obtener

comentarios de los primeros usuarios de forma rápida, sin invertir tiempo en funciones que podrían no ser necesarias.
- **Amplificar el aprendizaje:** Tras el lanzamiento inicial, el equipo analiza el comportamiento de los usuarios. Descubren que la mayoría de la gente prefiere un método de pago distinto al que implementaron. Usan esta información para priorizar la siguiente funcionalidad.
- **Potenciar al equipo:** En lugar de esperar a que un gerente tome todas las decisiones, el equipo de desarrollo decide de manera autónoma qué herramienta usar para gestionar las suscripciones, ya que tienen el conocimiento técnico para hacerlo

Crystal

Descripción: A diferencia de otras metodologías ágiles que siguen un conjunto de reglas rígidas, **Crystal** es una "familia" de metodologías. Su principio fundamental es que no existe un único método que funcione para todos los proyectos. En su lugar, se adapta al tamaño del equipo y a la criticidad del proyecto. Los dos enfoques más conocidos son **Crystal Clear** (para equipos pequeños) y **Crystal Orange** (para equipos más grandes).

- **Elementos clave:**
    - **Prioridad en las personas:** Crystal se enfoca en las interacciones y la comunicación del equipo. Cree que la comunicación cara a cara es el medio más eficaz para colaborar.
    - **Adaptabilidad:** No tiene un conjunto fijo de roles o eventos. En su lugar, el equipo decide qué reglas y prácticas funcionarán mejor para su proyecto.
    - **Comunicación frecuente:** Fomenta la comunicación constante y la entrega de software funcional en ciclos cortos, permitiendo que el equipo y el cliente se mantengan alineados.
    - **Reflexión periódica:** El equipo se reúne para reflexionar sobre su proceso y ajustarlo, similar a la retrospectiva de Scrum.

Método de Desarrollo de Sistemas Dinámicos (DSDM)

- **Descripción:** DSDM (Dynamic Systems Development Method) es una metodología ágil que se enfoca en la entrega rápida y continua de soluciones. Fue desarrollada en la década de 1990 para proporcionar un marco ágil más estructurado, especialmente para proyectos empresariales donde la entrega a tiempo y la calidad son cruciales.
- **Principios clave:** DSDM se rige por ocho principios fundamentales que guían el comportamiento del equipo:
    - **Enfocarse en la necesidad del negocio:** Cada decisión del proyecto debe alinearse con el valor que aportará al negocio.
    - **Entregar a tiempo:** El tiempo es un factor fijo y los requisitos son flexibles para asegurar que el proyecto se complete dentro del plazo.
    - **Colaborar:** La colaboración estrecha entre todas las partes interesadas (stakeholders, desarrolladores, usuarios) es fundamental para el éxito.
    - **Nunca comprometer la calidad:** La calidad del producto no es negociable y se mantiene durante todo el proceso.
    - **Construir incrementalmente a partir de cimientos firmes:** Se comienza con una base sólida y se añaden funcionalidades en incrementos.
    - **Desarrollar iterativamente:** El trabajo se divide en ciclos de desarrollo cortos para mejorar continuamente el producto.
    - **Comunicarse de forma continua y clara:** La comunicación constante evita malentendidos y mantiene a todos informados.
    - **Demostrar control:** El equipo debe ser transparente y mostrar el progreso del proyecto de manera clara y controlada.

- **Fases del ciclo de vida:** DSDM divide el ciclo de vida del proyecto en tres fases principales, que a su vez contienen sub-fases:
    - **Fase de pre-proyecto:** Se identifican los proyectos a realizar y se asegura que sean viables y necesarios para la empresa.
    - **Fase del ciclo de vida del proyecto:** Esta es la fase principal y se subdivide en varias etapas:
        - **Estudio de viabilidad y estudio del negocio:** Se evalúa si el proyecto es factible y se definen los requisitos de negocio a un alto nivel.

        - **Iteración del modelo funcional:** Se desarrollan prototipos y modelos para validar las funcionalidades con el cliente.
        - **Diseño e iteración de la estructura:** Se diseña la arquitectura del sistema y se construye el producto de manera iterativa.
        - **Implementación:** El producto se despliega en el entorno operativo y se capacita a los usuarios.
    - **Fase de post-proyecto:** Se revisa si los beneficios esperados del proyecto se han alcanzado.

- **Técnica de priorización (MoSCoW):** DSDM utiliza la técnica **MoSCoW** para priorizar los requisitos, lo cual es un elemento clave de su metodología:
    - **M - Must have (Debe tener):** Requisitos esenciales que el proyecto debe cumplir.
    - **S - Should have (Debería tener):** Requisitos importantes, pero que no son cruciales para el lanzamiento.
    - **C - Could have (Podría tener):** Requisitos deseables pero de menor impacto.
    - **W - Won't have (No tendrá por ahora):** Requisitos que se posponen para versiones futuras.

- ## Importancia y beneficios de las metodologías ágiles

  Independientemente de la metodología ágil que se elija, ya sea Scrum, Kanban, XP, Lean u otra, todas comparten una serie de beneficios fundamentales que las hacen tan valiosas en la gestión de proyectos modernos. Estos beneficios no solo optimizan el proceso de trabajo, sino que también mejoran la calidad del producto final y la satisfacción de todas las partes involucradas.

- **Flexibilidad y adaptabilidad**

  Las metodologías ágiles permiten a los equipos reaccionar de forma rápida y eficiente ante los cambios. En un mercado dinámico donde los requisitos pueden evolucionar, la capacidad de adaptarse es crucial. En lugar de seguir un plan rígido que puede quedar obsoleto, los equipos ágiles ajustan sus prioridades y su enfoque en ciclos cortos. Esto reduce significativamente el riesgo de invertir tiempo y recursos en un producto que ya no se alinea con las necesidades del cliente.

- **Entrega de valor temprana y frecuente**

  A diferencia de los métodos tradicionales, donde el cliente recibe el producto completo al final de un largo proceso, las metodologías ágiles se centran en la entrega de incrementos funcionales de forma constante. Esto significa que el cliente puede empezar a usar partes del producto mucho antes, proporcionando valor desde el principio. Esta entrega temprana también permite obtener retroalimentación valiosa que se puede utilizar para mejorar el producto en los siguientes ciclos de desarrollo.

- **Colaboración y transparencia**

  La colaboración constante es un pilar de las metodologías ágiles. Se fomenta la comunicación abierta y directa tanto dentro del equipo como con los clientes o stakeholders. Herramientas visuales como los tableros Kanban o los eventos de Scrum como el *Daily Scrum* y la *Sprint Review* garantizan que todos tengan una visión clara del progreso del proyecto. Esta transparencia fortalece la confianza y asegura que todos los involucrados estén alineados con los objetivos.

- **Mejora continua**

  Las metodologías ágiles integran la mejora continua en su núcleo. Al final de cada ciclo de trabajo (como un sprint en Scrum), los equipos dedican tiempo a reflexionar sobre su desempeño. En sesiones como la *Retrospective*, analizan qué salió bien, qué no y cómo pueden mejorar su proceso en el futuro. Este enfoque iterativo asegura que el equipo evolucione constantemente, volviéndose más eficiente, productivo y resolviendo problemas de forma proactiva.

- **Mayor calidad del producto**

  La alta calidad es un beneficio directo de la forma en que trabajan los equipos ágiles. Al centrarse en la entrega de pequeñas funcionalidades, estas se pueden probar y validar de manera exhaustiva en cada ciclo. Prácticas como la refactorización constante del código en XP o la eliminación de desperdicios en Lean garantizan que el producto final no solo cumpla con los requisitos, sino que también sea robusto, fácil de mantener y con menos errores.

METODOLOGÍAS ÁGILES

Son enfoques de gestión de proyectos que promuéven la flexibilidad. la colaboracion y ala entrega continua de valor.

SCRUM



Enfoque basado en iteraciones cortas y roles definidos

KANBAN



Visualiza el flujo de trabajo y limita el trabajo en curso

EXTREME XP



Ciclos cortos, desarrollo por pruebas

LEAN



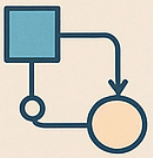
Elimina desperdicios y enfoca la mejora continua

CRYSTAL



Conjunto de metodologias ligeros y adaptables

DSDM



Desarrollo centrado en los requisitos y plazos establecidos



Featuree-Driven Development (FDD)

DSDM
Dynamic Systems

Webgrafía

- **Atlassian.** (s.f.). *¿Qué es la metodología ágil?*. Recuperado de https://www.atlassian.com/es/agile
- **IEBS Business School.** (2023). *Metodología Agile: ejemplos inspiradores para tu empresa*. Recuperado de https://interimgrouphr.com/blog/metodologia-agile-ejemplos/
- **SENA.** (2020). *¿Por qué usar Scrum?, como metodología de desarrollo de proyectos ágiles en el Sena*. Revista Integra. Recuperado de https://revistas.sena.edu.co/index.php/int/article/view/2686
- **Salesforce.** (s.f.). *Metodologías Ágiles: qué son y cómo pueden ayudarte*. Recuperado de https://www.salesforce.com/mx/blog/que-son-metodologias-agiles/

SENA