SERVICIO NACIONAL DE APRENDIZAJE SENA

Centro: Centro de Diseño e Innovación Tecnológica Industrial

Ficha: 3235906

Programa de Formación: TECNOLOGÍA EN ANÁLISIS Y DESARROLLO DE SOFTWARE

Nombre del Aprendiz: Laura Barona Saavedra

Nombre del Instructor: Felmaber Garzón Muñoz

Fecha: 7 de julio 2025

INTRODUCCIÓN

El desarrollo de software es una disciplina que requiere organización, planificación y metodologías que guíen el trabajo del equipo de desarrollo. Las metodologías de desarrollo de software permiten establecer procesos claros para la creación de productos tecnológicos eficientes y de calidad, adaptándose a diferentes contextos y necesidades del cliente.

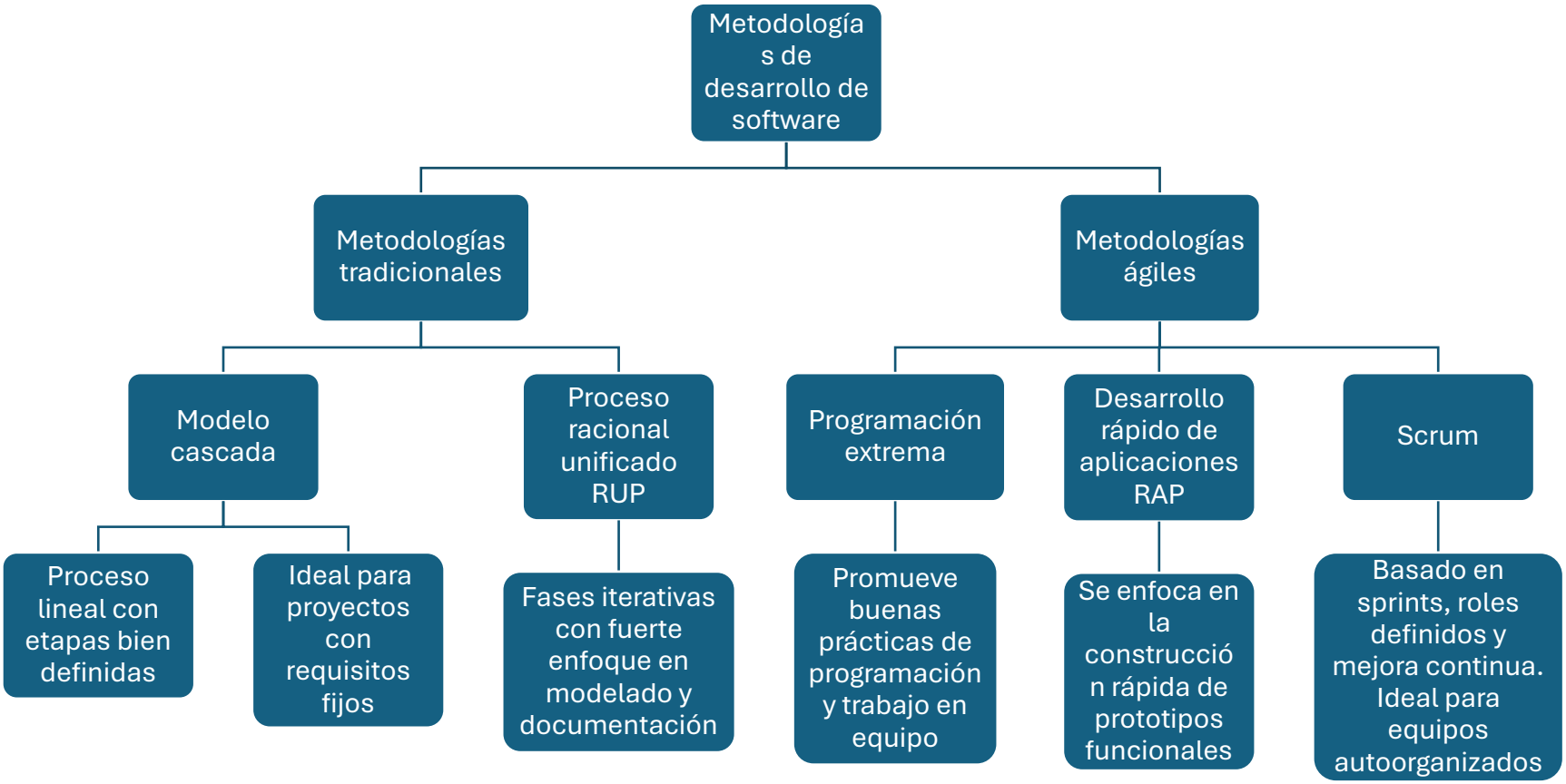
Este taller tiene como objetivo explorar las principales metodologías de desarrollo de software, sus clasificaciones, características, y la importancia de su aplicación en los proyectos. Se analizarán tanto marcos de trabajo tradicionales como ágiles, destacando sus ventajas, usos y enfoques, con base en el material de estudio proporcionado por el SENA y otras fuentes especializadas.

METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software son guías estructuradas que orientan al equipo de trabajo a lo largo del proceso de creación de un producto de software, desde la recolección de requisitos hasta su implementación y mantenimiento. Estas metodologías definen los procedimientos, roles, herramientas y principios necesarios para llevar a cabo un proyecto de manera organizada y eficaz.

Su principal objetivo es asegurar la calidad del producto final, optimizar los recursos disponibles, reducir los riesgos y promover la colaboración entre los actores del proyecto. La elección de una metodología adecuada depende del contexto, los requerimientos del cliente, el tamaño del equipo, la estabilidad de los requisitos y otros factores clave.

MAPA CONCEPTUAL



CLASIFICACIÓN DE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software se pueden clasificar en dos grandes grupos: **tradicionales** y **ágiles**, según su enfoque, estructura, capacidad de adaptación y participación del cliente en el proceso. Esta clasificación permite entender mejor cuándo es conveniente aplicar cada tipo, dependiendo del contexto del proyecto, la estabilidad de los requerimientos y la cultura del equipo.

Metodologías tradicionales

Estas metodologías siguen un **enfoque secuencial o estructurado**. Se basan en una planificación rigurosa desde el inicio del proyecto, donde se definen claramente los requerimientos, el diseño, la implementación y las pruebas. Son más efectivas cuando los requerimientos del software están completamente definidos desde el inicio y es poco probable que cambien.

Principales características:

- Fases rígidas y bien definidas.
- Alta documentación en cada etapa del proceso.
- Difícil adaptación a cambios inesperados.
- Bajo nivel de interacción con el cliente durante el desarrollo.
- Útiles en proyectos con bajo grado de incertidumbre o con regulaciones estrictas.

Ejemplos:

- **Modelo en cascada**: cada fase inicia una vez que la anterior ha concluido. Se centra en el cumplimiento de una secuencia estricta de pasos.

- **RUP (Proceso Racional Unificado)**: combina estructura y flexibilidad mediante iteraciones, pero mantiene una fuerte orientación a la documentación y a la arquitectura del sistema.

Metodologías ágiles

Estas metodologías surgieron como respuesta a la rigidez de los modelos tradicionales. Promueven la **flexibilidad, la comunicación constante y la entrega temprana de valor** al cliente. Están diseñadas para entornos cambiantes donde los requerimientos pueden evolucionar durante el desarrollo del producto.

Principales características:

- Trabajo iterativo e incremental (sprints).
- Fuerte colaboración entre el equipo y el cliente.
- Aceptación del cambio como parte del proceso.
- Énfasis en el software funcional más que en la documentación.
- Entregas frecuentes y mejora continua.

Ejemplos:

- **Scrum**: organización del trabajo en ciclos cortos con roles, eventos y artefactos definidos.
- **XP (Programación Extrema)**: prácticas de programación colaborativa, pruebas automatizadas y diseño simple.

- **RAD (Desarrollo Rápido de Aplicaciones)**: desarrollo veloz mediante la creación de prototipos y reutilización de componentes.

CARACTERÍSTICAS DE LOS MARCOS DE DESARROLLO DE SOFTWARE

Marcos de trabajo tradicionales

Los marcos tradicionales se caracterizan por ser estructurados, secuenciales y orientados al control del proceso mediante documentación y planificación rigurosa. A continuación, se describen sus características más relevantes:

Modelo en Cascada

- Enfoque secuencial: cada fase debe completarse antes de comenzar la siguiente.
- Fases definidas: requisitos → diseño → implementación → verificación → mantenimiento.
- Alta documentación: cada etapa deja como resultado documentos formales.
- Estabilidad de requisitos: ideal cuando los requerimientos son claros y poco cambiantes.
- Escasa retroalimentación del cliente: se involucra solo al final del proceso.
- Riesgo de detectar errores tarde: muchas fallas solo se identifican en etapas finales.
- Dificultad para implementar cambios a mitad del desarrollo.

Proceso Racional Unificado (RUP)

- Basado en iteraciones dentro de fases: inicio, elaboración, construcción y transición.
- Guiado por casos de uso: los requisitos se capturan mediante historias funcionales.

- Énfasis en la arquitectura: fuerte foco en el modelado del sistema (UML).
- Artefactos detallados: documentos de visión, requisitos, diagramas de clases, etc.
- Roles bien definidos: analistas, diseñadores, arquitectos, testers, stakeholders, etc.
- Disciplina y control: combina estructura del modelo en cascada con flexibilidad limitada.

Marcos de trabajo ágiles

Los marcos ágiles se enfocan en la adaptabilidad, la colaboración continua y la entrega temprana de valor al cliente. A diferencia de los tradicionales, promueven la flexibilidad frente al cambio y la mejora constante.

Scrum

- Trabajo por sprints: ciclos cortos de 2 a 4 semanas con entregables funcionales.
- Roles definidos: Product Owner, Scrum Master, equipo de desarrollo.
- Transparencia: todos pueden conocer el estado del proyecto (Scrum board, burndown chart).
- Inspección y adaptación: reuniones diarias, de revisión y retrospectiva.
- Priorización de valor: el Product Backlog se ordena por valor para el cliente.
- Mejora continua: cada sprint incluye una retrospectiva para identificar ajustes.

Programación Extrema (XP)

- Iteraciones cortas y frecuentes.

- Pruebas automatizadas: enfoque en TDD (Test-Driven Development).
- Refactorización constante: mejora del código sin cambiar su funcionalidad.
- Desarrollo en pareja (pair programming): dos programadores trabajan juntos en una sola máquina.
- Comunicación continua: uso de pizarras y conversaciones cara a cara.
- Entregas frecuentes: software funcional en cada iteración.

Desarrollo Rápido de Aplicaciones (RAD)

- Prototipado rápido: construcción de prototipos funcionales desde el inicio.
- Involucra al usuario desde el principio: validación constante.
- Iteraciones breves: se avanza por módulos de funcionalidad.
- Reutilización de componentes: se aprovecha código existente para agilizar entregas.
- Equipos multidisciplinarios: diseñadores, programadores, facilitadores, usuarios finales.

IMPORTANCIA DE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Implementar una metodología adecuada en un proyecto de software tiene múltiples beneficios:

- **Organización del trabajo**: se define un marco claro que facilita la coordinación entre los miembros del equipo.
- **Calidad del producto**: se aplican prácticas que aseguran la funcionalidad, usabilidad y rendimiento del software.

- **Gestión de tiempos y recursos**: se puede estimar de manera más precisa el esfuerzo y duración del proyecto.
- **Reducción de riesgos**: permite identificar y mitigar posibles problemas desde etapas tempranas.
- **Satisfacción del cliente**: especialmente en metodologías ágiles, el cliente recibe entregas frecuentes y participa en la toma de decisiones.

La correcta elección e implementación de una metodología incrementa considerablemente las posibilidades de éxito del proyecto y fomenta una cultura de mejora continua.

CONCLUSIÓN

Las metodologías de desarrollo de software representan una herramienta clave para asegurar el éxito en la creación de productos tecnológicos. A través de su correcta aplicación, los equipos de trabajo logran organizar y estructurar el proceso de desarrollo, optimizando el uso de recursos, tiempos y esfuerzos, y garantizando la calidad del producto final.

La clasificación en metodologías tradicionales y ágiles permite seleccionar el enfoque más adecuado según las características del proyecto. Mientras las metodologías tradicionales ofrecen una estructura rígida y detallada, ideales para proyectos con requerimientos estables, las metodologías ágiles brindan flexibilidad y adaptabilidad, siendo más efectivas en entornos dinámicos y cambiantes.

Conocer las características de cada marco de trabajo —como el modelo en cascada, RUP, Scrum, XP o RAD— permite a los desarrolladores tomar decisiones más informadas, implementar buenas prácticas y adaptarse mejor a las necesidades del cliente y del equipo. Además, promueve una cultura de mejora continua y colaboración, elementos esenciales en la industria del software actual.

En definitiva, aplicar una metodología de desarrollo no es solo una cuestión técnica, sino una estrategia fundamental para construir soluciones tecnológicas eficientes, sostenibles y centradas en el usuario.

REFERENCIAS

SENA. (s.f.). *Metodologías de desarrollo de software* [Material de clase, documento proporcionado por la institución].