



FACULTAD DE INDUSTRIAS AGROPECUARIAS Y CIENCIAS
AMBIENTALES

NORMATIVAS DE INGENIERÍA DE SOFTWARE

Tema: Metodologías de
Desarrollo de Software

Fecha: 23-02-2025

Integrantes: Geovanny Basantes

Docente: Msc. Jorge Miranda

Informe sobre Metodologías de Desarrollo de Software

1. Introducción

En el ámbito de la ingeniería de software y en la disciplina del desarrollo de aplicaciones informáticas, es fundamental contar con **procesos estructurados y metodológicamente definidos** para garantizar la correcta planificación, ejecución, monitoreo y control de los diferentes proyectos de software que se desarrollan en la actualidad. Estos procesos deben permitir una ejecución óptima, minimizando los errores y maximizando la eficiencia en términos de recursos computacionales, humanos y temporales.

A lo largo del tiempo, han surgido diversas **metodologías de desarrollo de software**, cada una con su propio enfoque, objetivos específicos y conjunto de principios fundamentales que guían el desarrollo de productos informáticos. Dichas metodologías se pueden clasificar, de manera general, en dos grandes categorías:

1. **Metodologías Tradicionales o Planificadas:** Enfoques estructurados y secuenciales que siguen un orden predefinido de etapas, en donde cada fase debe completarse antes de avanzar a la siguiente.
2. **Metodologías Ágiles o Iterativas:** Modelos más flexibles y adaptativos que permiten cambios y ajustes sobre la marcha, priorizando la entrega continua de software funcional y la retroalimentación constante con los usuarios.

Este informe detallado tiene como finalidad analizar a fondo ambas categorías de metodologías, comparando sus características, ventajas, desventajas y áreas de aplicación, para proporcionar un panorama amplio y profundo sobre la selección adecuada de metodologías en el contexto del desarrollo de software.

2. Metodologías Tradicionales de Desarrollo de Software

Las **metodologías tradicionales**, también denominadas **metodologías clásicas o secuenciales**, están basadas en un enfoque predecible y estructurado, en donde cada fase del desarrollo de software se lleva a cabo en un orden lógico y lineal. Dentro de este enfoque, se encuentran modelos como el **modelo en cascada**, el **modelo espiral**, el **modelo V** y **otros enfoques estructurados** ampliamente utilizados en entornos empresariales con requerimientos bien definidos.

2.1 Modelo en Cascada (Waterfall Model)

El **modelo en cascada**, denominado en inglés *Waterfall Model*, es uno de los enfoques más antiguos y tradicionales en el desarrollo de software. Su estructura está basada en un **flujo de trabajo secuencial y progresivo**, en donde cada fase del desarrollo debe completarse en su totalidad antes de avanzar a la siguiente.



Ilustración 1 Metodología Cascada

Las fases principales del modelo en cascada incluyen:

1. **Recolección y análisis de requisitos:** Se definen detalladamente los requerimientos del sistema.
2. **Diseño del sistema:** Se diseña la arquitectura general del software a desarrollar.
3. **Implementación y codificación:** Se escriben las líneas de código necesarias para el desarrollo del software.

4. **Pruebas y verificación:** Se realizan pruebas exhaustivas para detectar y corregir errores.
5. **Despliegue y mantenimiento:** El software es implementado y se realizan mantenimientos periódicos.

Ventajas del Modelo en Cascada:

- Proporciona un **enfoque estructurado y organizado** en el desarrollo.
- Permite una planificación clara desde el inicio del proyecto.
- Es útil para proyectos con requerimientos bien definidos y poco cambiantes.

Desventajas del Modelo en Cascada:

- No permite cambios en los requerimientos una vez iniciada la fase de desarrollo.
- La detección de errores suele ocurrir en etapas avanzadas, generando costos elevados.

2.2 Modelo Espiral

El **modelo espiral**, desarrollado por Barry Boehm **【10】** , combina elementos del desarrollo secuencial con características iterativas. Su principal característica es que integra el **análisis de riesgos en cada una de sus fases**, permitiendo realizar ajustes en cada iteración del desarrollo.

Este modelo sigue un ciclo compuesto por las siguientes etapas:

1. **Planificación:** Definición de objetivos, alcance y restricciones del software.
2. **Análisis de riesgos:** Identificación de posibles problemas y estrategias de mitigación.
3. **Desarrollo e implementación:** Construcción de prototipos incrementales y mejoras progresivas.
4. **Evaluación y validación:** Revisión y validación de resultados con los clientes y usuarios finales.

Ventajas del Modelo Espiral:

- Permite la incorporación de cambios de manera controlada.
- Ofrece un enfoque iterativo con validaciones constantes.

Desventajas del Modelo Espiral:

- Requiere una planificación detallada en cada ciclo.
 - Puede aumentar la complejidad del proyecto si no se gestiona adecuadamente.
-

3. Metodologías Ágiles de Desarrollo de Software

En contraposición a los modelos tradicionales, las **metodologías ágiles** han ganado gran popularidad debido a su enfoque basado en la entrega iterativa, la flexibilidad ante cambios y la colaboración activa con los clientes y usuarios finales.

3.1 Scrum

El **Scrum** es una metodología ágil que se basa en la organización del trabajo en ciclos cortos llamados **sprints**, donde se entrega software funcional al final de cada iteración.

Roles principales en Scrum:

- **Scrum Master:** Facilita la metodología y elimina impedimentos del equipo.
- **Product Owner:** Representa los intereses del cliente y prioriza funcionalidades.
- **Equipo de desarrollo:** Conforman los programadores y testers encargados de la implementación.

Ventajas de Scrum:

- Permite una **entrega rápida y frecuente de software** funcional.
- Mejora la **colaboración y comunicación** dentro del equipo de desarrollo.

Desventajas de Scrum:

- Puede ser difícil de implementar en equipos no acostumbrados a metodologías ágiles.

3.2 Kanban

El **Kanban** es un enfoque visual basado en tableros que permiten gestionar el flujo de trabajo de manera continua. Se enfoca en la reducción de tiempos de espera y la optimización de la eficiencia en los procesos de desarrollo.

Ventajas de Kanban:

- Brinda una **visión clara del progreso del proyecto**.
- Reduce los **cuellos de botella en el desarrollo**.

Desventajas de Kanban:

- No proporciona una estructura rígida de trabajo.

4. Comparación entre Metodologías Tradicionales y Ágiles

Criterio	Metodologías Tradicionales	Metodologías Ágiles
Flexibilidad	Baja	Alta
Documentación	Extensa	Reducida
Entrega	Al final del proyecto	Iterativa y continua
Adaptabilidad	Baja	Alta

5. Conclusión y Reflexión Final

Las metodologías de desarrollo de software son **herramientas esenciales para la planificación, ejecución y control de proyectos informáticos**. La elección entre una metodología tradicional o ágil dependerá de la naturaleza del proyecto, la cultura del equipo y las necesidades del cliente. Mientras que los enfoques tradicionales son adecuados para proyectos con requisitos bien definidos, las metodologías ágiles han demostrado ser más efectivas en entornos dinámicos y cambiantes.

En consecuencia, es fundamental que los equipos de desarrollo **seleccionen la metodología que mejor se adapte a sus necesidades específicas**, considerando los beneficios y limitaciones de cada enfoque.

Bibliografía:

Maida, E. G., & Pacienza, J. (2015). Metodologías de desarrollo de software.

Gómez, O. T., López, P. P. R., & Bacalla, J. S. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial data*, 13(2), 70-74.

Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista de Tecnología e Innovación*, 2(5), 980-986.

Duarte, A. O., & Rojas, M. (2008). Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. *Revista Avances en Sistemas e Informática*, 5(2), 159-171.