Ciclo de Vida del Software (SDLC)

1. ¿Qué es el Ciclo de Vida del Software? (SDLC)

🔍 El ciclo de vida del software es un **proceso que abarca desde la concepción hasta la finalización** de un sistema o aplicación.

🎯 **Objetivo:** Asegurar que el software cumpla con los requisitos y objetivos establecidos.

⚙️ **SDLC (Software Development Life Cycle):** Describe las fases necesarias para validar el desarrollo y garantizar la calidad.

YouTube

Ciclo de vida desarrollo de software | Qué es el desarrollo de software

El ciclo de vida del software (SDLC) nos permite identificar, administrar y planificar la gestión de recursos hasta alcanzar un objetivo propuesto. La cantidad de fases que...

Referencias: StarkCloud | AWS | ServiceNow

2. Fases del Ciclo de Vida del Software

1. Planificación 📝

Definición de objetivos, identificación de requisitos y creación del plan de desarrollo.

1

2. Análisis 🔎

Comprensión profunda de las necesidades del usuario y los requerimientos técnicos.

2

3. Diseño 🏗️

Especificación de la arquitectura y diseño técnico del sistema.

3

4. Implementación (Codificación) 💻

Traducción de los diseños en código funcional.

4

5. Pruebas 🧪

Identificación y corrección de errores para garantizar calidad.

5

6. Despliegue 🚀

Instalación del software en el entorno de producción para los usuarios finales.

6

7. Mantenimiento 🔄

Actualización continua para corregir fallos y añadir nuevas funcionalidades.

7

3. Modelos del Ciclo de Vida del Software

Los modelos ayudan a **organizar las fases** y a **gestionar proyectos** según sus características.

Modelo en Cascada 🌊

🔎 **Características:**

- Modelo secuencial y lineal.
- Cada fase debe completarse antes de pasar a la siguiente.
- Adecuado para proyectos con requisitos bien definidos.

⚠️ **Desventajas:** ❌ Poca flexibilidad para cambios. Retroalimentación tardía (al final del proyecto).

Caso de Uso:

✅ Proyectos con requisitos claros y bien definidos desde el inicio.
Cuando no se esperan cambios significativos durante el desarrollo.
✅ Ideal para industrias con normativas estrictas que requieren documentación detallada.

📝 **Ejemplos Prácticos:**

- **Desarrollo de software para cajeros automáticos (ATM):** Los requisitos son estables y la seguridad es crítica.
- **Sistemas de control de tráfico aéreo:** Cambios durante el desarrollo son costosos y riesgosos.
- **Proyectos gubernamentales:** Requieren documentación exhaustiva y procesos predecibles.
- **Aplicaciones para hardware específico:** Como controladores de impresoras o firmware de electrodomésticos.

Modelo en Espiral 🌀

1 Características:

✅ Basado en la gestión de riesgos.

Iterativo: Repite fases para mejorar el producto.

Incluye planificación, análisis de riesgos, desarrollo y evaluación.

2 Ventajas:

✔️ Alta flexibilidad.

Permite prototipos y retroalimentación temprana.

Minimiza riesgos.

3 Desventajas:

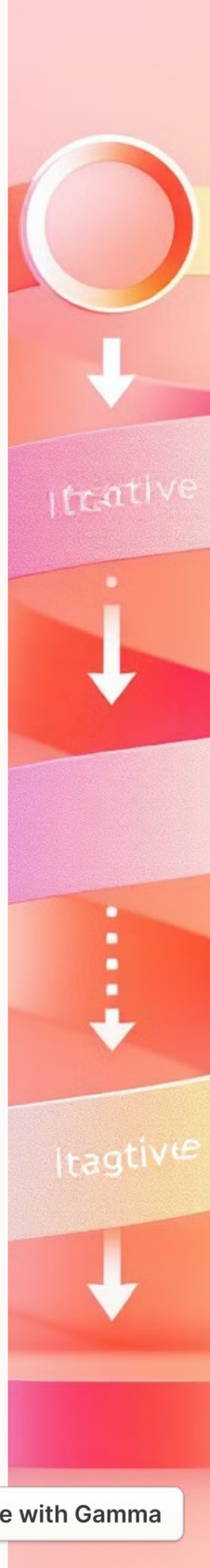
❌ Puede ser costoso.
Requiere experiencia en gestión de riesgos.

4 🔎 Caso de Uso:

✅ **Proyectos grandes, complejos y de alto riesgo.**
Cuando los requisitos pueden cambiar durante el desarrollo.
✅ **Ideal cuando es necesario realizar evaluaciones de riesgos frecuentes.**

📝 **Ejemplos Prácticos:**

- **Desarrollo de sistemas financieros:** Como plataformas bancarias que requieren alta seguridad y gestión de riesgos.
- **Sistemas de gestión empresarial (ERP) a medida:** Permite la evolución basada en la retroalimentación del cliente.
- **Desarrollo de videojuegos AAA:** Requiere múltiples pruebas y validaciones antes del lanzamiento final.
- **Proyectos de defensa y aeroespaciales:** Donde la gestión de riesgos es fundamental.



Modelo Iterativo 🔄

Características:

✅ Desarrollo en ciclos repetitivos.

Cada iteración produce una versión mejorada del software.

Permite retroalimentación y ajustes constantes.

Ventajas:

🚀 ✔️ Flexibilidad ante cambios.
Entregas incrementales y más rápidas.
Menor riesgo de fallos finales.

Desventajas:

⚠️ ❌ Requiere gestión constante.
Puede ser difícil predecir plazos finales.

🔎 **Caso de Uso:**
✅ Cuando se necesita lanzar versiones funcionales rápidamente.
Proyectos donde la retroalimentación del usuario es esencial para mejorar.
✅ Ideal para proyectos con cambios frecuentes en los requisitos.

📝 **Ejemplos Prácticos:**

- **Aplicaciones móviles y web:** Como redes sociales o apps de mensajería que se actualizan regularmente.
- **Software de comercio electrónico:** Permite lanzar funcionalidades nuevas mientras se mejora la experiencia del usuario.
- **Plataformas educativas en línea:** Se ajustan según la retroalimentación de los estudiantes y profesores.
- **Herramientas de productividad (como Google Docs o Trello):** Se actualizan constantemente con nuevas características.

4. Comparativa entre Modelos

Modelo	Caso de Uso Ideal	Ejemplo Real
Cascada	Requisitos estables y proyectos bien definidos.	Sistema para tarjetas de crédito.
Espiral	Proyectos complejos con alta gestión de riesgos.	Software para misiones espaciales.
Iterativo	Proyectos con cambios constantes y entregas rápidas.	Aplicaciones móviles como WhatsApp.

5. ¿Cómo elegir el modelo adecuado? 🤔

Aspecto	Modelo en Cascada	Modelo en Espiral	Modelo Iterativo
Enfoque	Lineal y secuencial	Basado en riesgos	Repetitivo e incremental
Flexibilidad	Baja	Alta	Alta
Coste	Bajo (proyectos simples)	Alto (por evaluaciones constantes)	Moderado
Retroalimentación	Final del proceso	Continua	Continua
Uso Ideal	Requisitos estables	Proyectos complejos con riesgos	Cambios frecuentes y evolución rápida

Modelo en Cascada:

✔️ Proyectos pequeños y con requisitos fijos.

Documentación detallada desde el inicio.

Modelo en Espiral:

Proyectos grandes y complejos.

Requieren evaluación continua y gestión de riesgos.

Modelo Iterativo:

Proyectos con requisitos cambiantes.

✔️ Necesidad de entregar versiones funcionales rápidamente.

6. Metodologías Ágiles y el Ciclo de Vida

⚡ ¿Cómo se relacionan?

Las metodologías ágiles, como **Scrum** y **Kanban**, adoptan el modelo iterativo para entregar productos en ciclos cortos, fomentando la colaboración y adaptación.

🚀 Ventajas de lo ágil:

✔️ Adaptación constante a cambios.

Entrega rápida de valor al cliente.

✔️ Equipos autogestionados y colaborativos.

7. Conclusión 🏆

1. ✅ **El SDLC es esencial** para garantizar que el software cumpla con los objetivos y requisitos establecidos.
2. La **elección del modelo adecuado** depende del tipo de proyecto, recursos y cambios esperados.
3. **Modelos iterativos y ágiles** ofrecen flexibilidad, mientras que el modelo en cascada brinda estructura.
4. ✅ El **modelo en espiral** es ideal para proyectos con alta incertidumbre y necesidad de minimizar riesgos.