UNIVERSIDAD DE GUADALAJARA
Red Universitaria e Institución Benemérita de Jalisco

CUCEI
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Centro Universitario de Ciencias Exactas e Ingenierías

Materia: Ingeniería en Software

Universidad de Guadalajara

Itzel Paulina Esquivel Urenda

Juan Pablo González Riveras

Luis Fernando Lupercio Ramirez

La elección de una metodología adecuada para el desarrollo de software es un factor crucial que puede determinar el éxito o fracaso de un proyecto. Cada metodología ofrece un enfoque distinto para abordar las etapas del desarrollo, desde la especificación de requerimientos hasta la entrega final del producto. Estas metodologías se han diseñado para adaptarse a diferentes contextos, tipos de proyectos y necesidades específicas de los clientes.

Este documento tiene como objetivo presentar una tabla comparativa de tres metodologías de desarrollo de software ampliamente utilizadas: el Modelo en Cascada, el Desarrollo Incremental y la Ingeniería de Software Orientada a la Reutilización. Se analizan sus principales ventajas y desventajas, con el fin de ofrecer una visión clara y práctica que facilite la toma de decisiones en la selección del enfoque más adecuado para cada caso

Modelo en Cascada

Características:

Es un modelo secuencial lineal, donde cada fase debe completarse completamente antes de avanzar a la siguiente.

Se divide en fases claras y definidas:

1. Especificación de Requisitos: Se documentan los requisitos completos del sistema.
2. Diseño del Sistema y del Software: Se diseña la arquitectura del sistema y los componentes detallados.
3. Implementación y Pruebas Unitarias: Se codifican los módulos y se prueban de forma individual.
4. Integración y Pruebas del Sistema: Se integran los componentes y se valida el sistema completo.
5. Despliegue y Mantenimiento: Se entrega el producto al cliente y se realizan correcciones o actualizaciones.

Desarrollo Incremental

Características:

1. El sistema se desarrolla y entrega en partes (incrementos), cada uno funcional y construido sobre los anteriores.
2. Los requisitos iniciales no necesitan estar completamente definidos; pueden evolucionar con el tiempo.
3. Cada incremento pasa por el ciclo de especificación, diseño, implementación y prueba.

Ingeniería de Software Orientada a la Reutilización

Características:

1. Construcción de sistemas reutilizando componentes de software ya existentes.
2. Los componentes pueden ser librerías, frameworks o sistemas comerciales (COTS, por sus siglas en inglés).
3. En lugar de diseñar desde cero, el enfoque principal es integrar y personalizar.

	Ventajas	Desventajas
Modelo en Cascada	• **Simplicidad**: Fácil de entender y usar debido a su naturaleza estructurada. • **Planificación clara**: Permite definir hitos claros y específicos en cada fase. • Adecuado para proyectos con requisitos bien definidos: Funciona mejor cuando los requisitos son conocidos desde el inicio y poco susceptibles a cambios. • Facilita la documentación: Cada etapa produce documentación detallada, útil para el mantenimiento. • Es un proceso bien organizado que hace más sencilla la gestión del proyecto y la documentación. • Funciona muy bien cuando los requisitos del proyecto son claros y no van a cambiar.	• **Entrega tardía**: El cliente no ve resultados hasta el final del proyecto. • **Poco adaptable**: No es muy flexible, ya que, si los requisitos cambian durante el desarrollo, es complicado adaptarse. • Las etapas son muy rígidas, lo que puede generar problemas en proyectos donde hay mucha incertidumbre. • Si hay errores o cambios necesarios en etapas avanzadas, arreglarlos puede ser muy costoso por el orden lineal del proceso.

	• Como sigue un orden secuencial, es fácil ver en qué etapa del desarrollo se encuentra el proyecto.	
Desarrollo Incremental	• **Menor riesgo**: Los problemas se identifican y corrigen más fácilmente en incrementos pequeños. • Es más flexible porque permite hacer ajustes en cualquier momento durante el desarrollo. • Se puede recibir retroalimentación constante del cliente, lo que ayuda a cumplir mejor con lo que realmente necesita. • Las funcionalidades básicas pueden entregarse rápido, lo que aporta valor al cliente desde temprano.	• Si no se gestiona bien, las iteraciones pueden causar que la estructura del sistema se desorganice con el tiempo. • Requiere una buena planificación para que los incrementos estén bien coordinados y no choquen entre sí. • No es tan útil en proyectos donde todos los requisitos tienen que estar definidos desde el principio.
Ingeniería de Software Orientada a la Reutilización	• Ayuda a reducir mucho los costos y el tiempo al usar componentes que ya existen. • Se disminuyen riesgos porque se utilizan elementos que ya han sido probados y son confiables. • Promueve un uso más eficiente de los recursos al estandarizar ciertos componentes.	• Dependencia de los componentes disponibles: Si no se encuentran componentes adecuados, el proyecto puede retrasarse. • Limitaciones de personalización: Algunos componentes pueden no satisfacer completamente los requisitos.

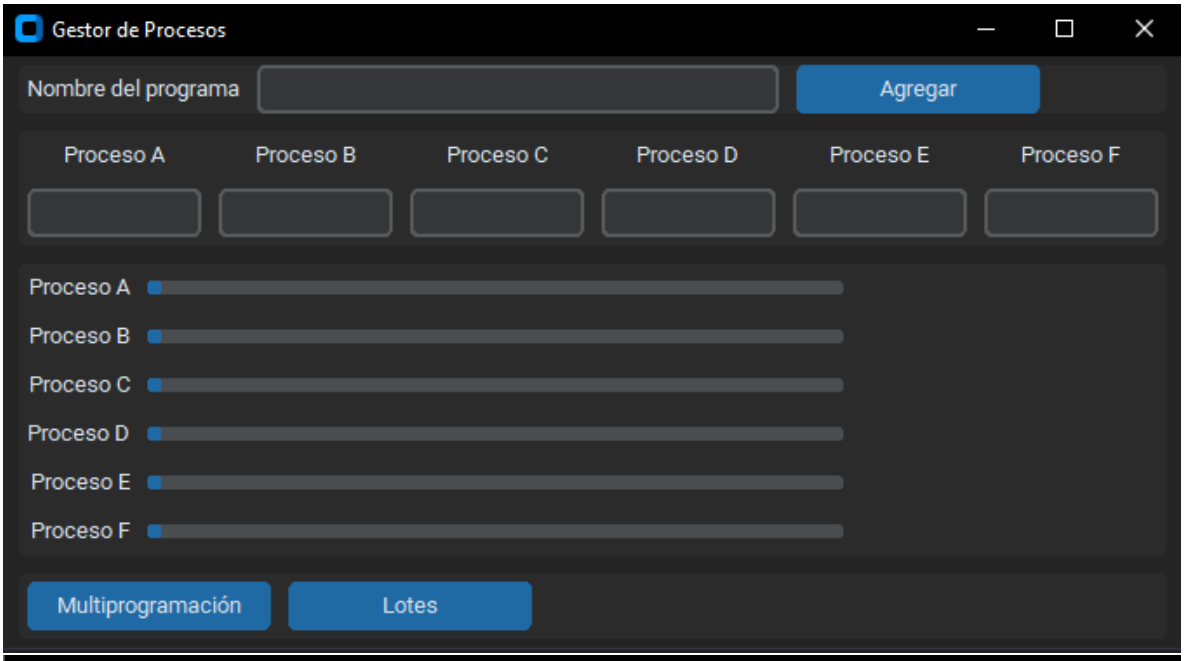
		• Costos de licencia: Si se utilizan componentes comerciales, estos pueden tener un costo significativo. • Puede ser difícil personalizar completamente, ya que los componentes reutilizados podrían no ajustarse 100% a las necesidades específicas del proyecto. • Depender de elementos externos puede generar problemas cuando se necesiten actualizaciones o mantenimiento. • Hay menos control sobre cómo evoluciona el sistema a largo plazo. •

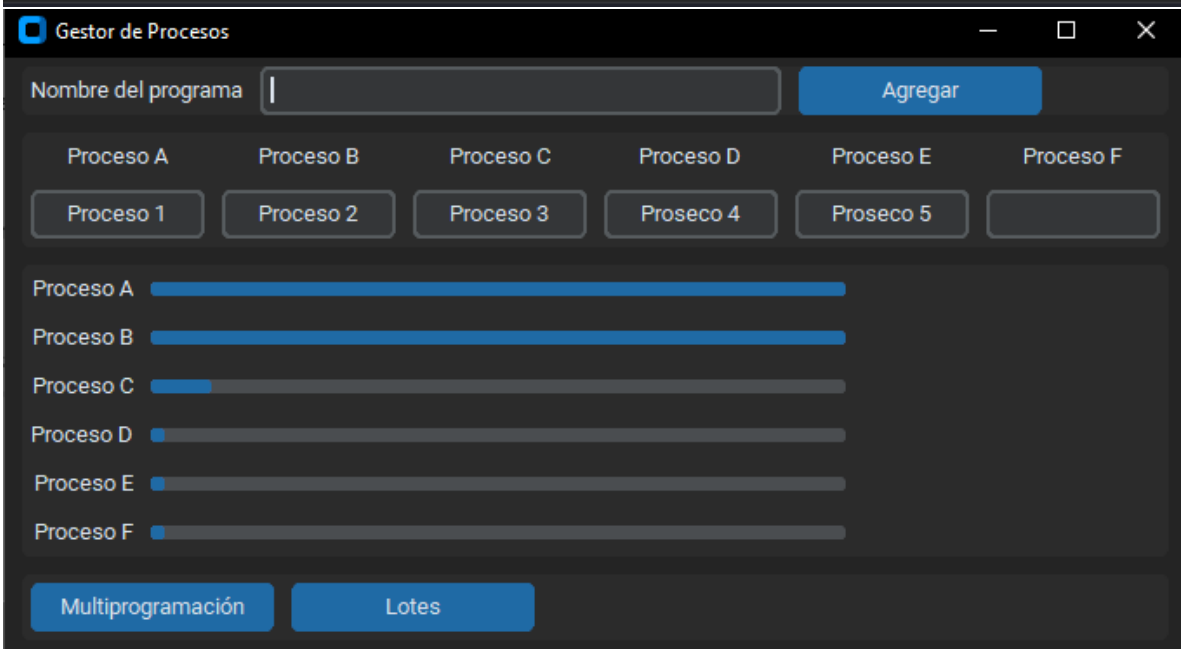
Comparación General entre los Modelos

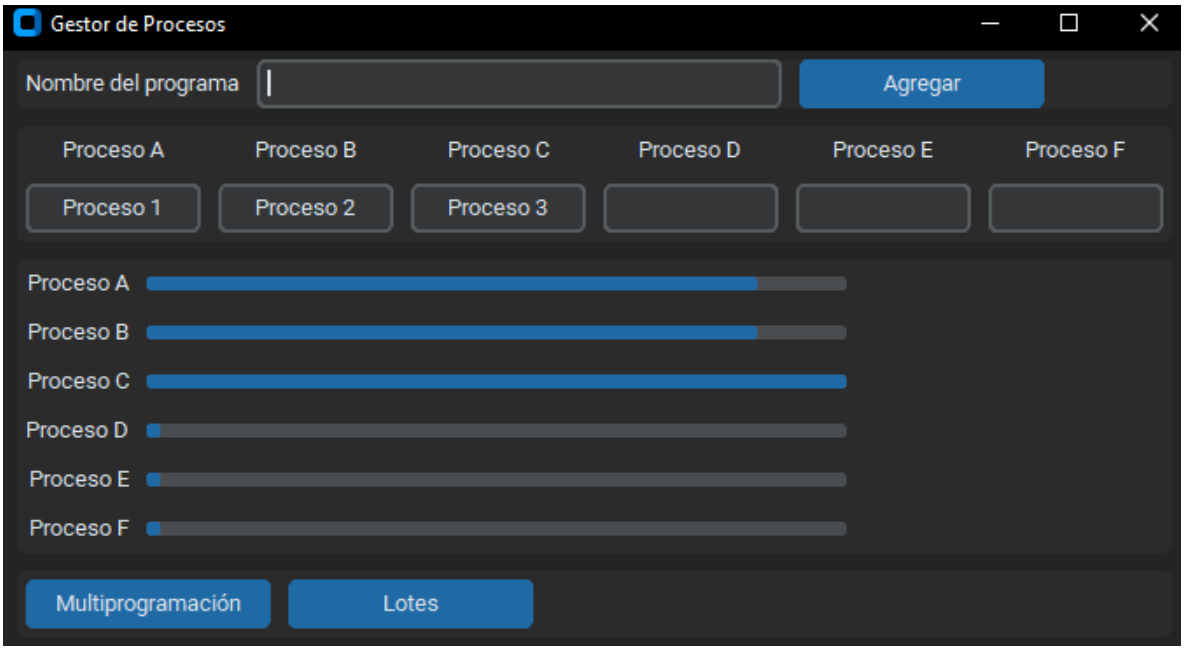
Aspecto	Modelo de cascada	Desarrollo incremental	Ingeniería de Software Orientada a la Reutilización
Estructura	Secuencia y rígida	Iterativa y flexible	Basado en integración de componentes
Entrega al cliente	Al final del proyecto	Gradual con cada incremento	Al integrar los componentes reutilizados
Adaptabilidad	Baja	Alta	Depende de la disponibilidad de componentes
Complejidad de gestión	Moderada	Alta	Alta debido a la integración
Requerimientos iniciales	Totalmente definidos	Pueden evolucionar	Basado en componentes disponibles
Costos	Altos ante cambios tardíos	Menores en cambios progresivos	Reducidos si los componentes son gratuitos

PROGRAMA DE LOTES O MULTIPROGRAMACIÓN

Además, como parte de esta tarea, se desarrolló un programa anclado en la sección inferior de Google Classroom. Este programa permite agregar hasta seis procesos simultáneamente, los cuales se pueden gestionar y ejecutar en lotes o bajo un esquema de multiprogramación.







Link del código:
https://github.com/Fernando9206/Actividad-de-aprendizaje-1.2-Tabla-Comparativa-Metodolog-as-de-Desarrollo-de-Software-.git

CONCLUCION:

La elección de una metodología para desarrollar software depende mucho de lo que necesite el proyecto, los recursos que tengas disponibles y el entorno donde se va a trabajar. Por ejemplo, el Modelo en Cascada funciona bien si los requisitos del proyecto ya están claros desde el principio. En cambio, el Desarrollo Incremental es mejor cuando el entorno es más dinámico y los cambios son frecuentes. Por su parte, la Ingeniería de Software Orientada a la Reutilización ayuda a ahorrar tiempo y dinero al usar componentes ya existentes, aunque esto puede hacer que el producto final sea menos personalizable.

REFERENCIAS

Arlow, J. y Neustadt, I. (2005). UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition). Boston: Addison-Wesley. Boehm, B. y Turner, R. (2003). Balancing Agility and Discipline: A Guide for the Perplexed. Boston: Addison-Wesley.

Boehm, B. W. (1988). “A Spiral Model of Software Development and Enhancement”. IEEE Computer, 21 (5), 61–72. Budgen, D. (2003). Software Design (2nd Edition). Harlow, UK.:

Addison-Wesley. Krutchen, P. (2003). The Rational Unified Process—An Introduction (3rd Edition). Reading, MA: Addison-Wesley. Massol, V. y Husted, T. (2003). JUnit in Action.

Greenwich, Conn.: Manning Publications Co. Rettig, M. (1994). “Practical Programmer: Prototyping for Tiny Fingers”. Comm. ACM, 37 (4), 21–7. Royce, W. W. (1970). “Managing the Development of Large Software Systems: Concepts and Techniques”.

IEEE WESTCON, Los Angeles CA: 1–9. Rumbaugh, J., Jacobson, I. y Booch, G. (1999). The Unified Software Development Process. Reading, Mass.: Addison-Wesley. Schmidt, D. C. (2006). “Model-Driven Engineering”.

IEEE Computer, 39 (2), 25–31. Schneider, S. (2001). The B Method. Houndmills, UK: Palgrave Macmillan. Wordsworth, J. (1996). Software Engineering with B. Wokingham: Addison Wesley