UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE

SEDE SANTO DOMINGO DE LOS TSÁCHILAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PERIODO	:	202350 Mayo 2023 – Septiembre 2023
ASIGNATURA	:	Metodología de desarrollo de Software
TEMA	:	Tarea
ESTUDIANTES	:	Sahid Bosquez, Juan Jimenez
NIVEL-PARALELO - NRC:		Tercer Semestre - 16132
DOCENTE	:	Ing. Javier Jose Cevallos Farias
FECHA DE ENTREGA	:	18/12/2023

Contenido

Ciclos de vida de un software

Introducción:

El ciclo de vida del desarrollo de software (SDLC) es un proceso rentable y eficiente en términos de tiempo empleado por los equipos de desarrollo para garantizar que el software sea fiable, seguro y responda a las necesidades de los usuarios finales. El SDLC consta de varias fases, que pueden variar según el modelo utilizado, pero generalmente incluyen: análisis de requisitos, diseño, implementación, pruebas y mantenimiento. Los modelos SDLC más conocidos son el modelo cascada, el modelo iterativo y el modelo ágil. El objetivo final del SDLC es lograr la satisfacción del cliente.

Estudios de Caso o Ejemplos Prácticos: Incluye ejemplos del mundo real para ilustrar las metodologías.

Ciclos de vida de un software:

Son como menciona su nombre el ciclo de vida para el desarrollo de un software por el cual todos los softwares deben pasar para su correspondiente funcionamiento, además el ciclo de vida de software está mayoritariamente diseñado para el desarrollo en grupos de trabajo para poder realizar sus diferentes funciones para culminar el desarrollo del software además de darle su respectivo mantenimiento hasta la finalización de este.

El ciclo de vida del desarrollo de software (SDLC) es un proceso rentable y eficiente en términos de tiempo empleado por los equipos de desarrollo para diseñar y crear software de alta calidad. El objetivo del SDLC es minimizar los riesgos del proyecto por medio de una planificación anticipada que permita que el software cumpla las expectativas del cliente durante la fase de producción y posteriormente. Esta metodología establece una serie de pasos que dividen el proceso de desarrollo de softwar en tareas que se pueden asignar, completar y medir.(Amazon, 2023).

Metodologías de ciclo de vida:

Las metodologías del ciclo de vida de un software son aquellas tipos de modelos para desarrollar un software los cuales tienen pasos para seguir en los cuales podemos encontrar:

Según (Solera, 2022) nos explica que los tipos de metodologías de software son:

Metodologías Ágiles:

- **Metodología Kanban.** Se enfoca en la mejora del flujo de trabajo y en la entrega de software de alta calidad. Nació en la industria automotriz japonesa en los años cuarenta y su nombre deriva de la palabra ‘tablero’, que se utiliza para visualizar el flujo de trabajo.
- **Metodología Scrum.** Es una metodología de desarrollo de software orientada a la entrega de productos de alta calidad a través de un proceso iterativo e incremental. En Scrum, un equipo de desarrollo se divide en equipos de trabajo autónomos que se

encargan de desarrollar una parte del producto. Cada equipo se reúne regularmente para revisar el progreso y planificar el trabajo futuro.

Metodologías Tradicionales:

Metodología de Cascada:

Es una forma de desarrollo de software en la que se siguen una serie de pasos estrictos y en orden. Esta metodología es muy detallada y deja poco margen de maniobra para el equipo de desarrollo. A menudo, se usa en proyectos grandes y complejos en los que es importante que todos los pasos se cumplan de forma estricta.

Metodología DevOps:

Destaca la colaboración estrecha entre el desarrollo y la operación, y es ideal para proyectos de software que requieren un ciclo de lanzamiento rápido. En definitiva, es una metodología orientada a la colaboración y coordinación entre el personal de desarrollo y el de operaciones, con el objetivo de mejorar la calidad y la velocidad de las entregas de software.

Metodología Lean:

Se enfoca en la minimización de desperdicios y en la entrega de software de alta calidad. Se basa en la filosofía Lean de la manufactura y se caracteriza por tener un enfoque en el cliente, en la minimización de desperdicios y en la mejora continua. Se basa en el principio de "entregar lo más valioso al cliente lo antes posible". El objetivo de Lean es mejorar la calidad y la eficiencia del producto o servicio, y reducir el tiempo y el costo de producción.

Metodología de Espiral:

Se basa en un enfoque iterativo e incremental para el desarrollo de software. Se divide en cuatro fases: Inicio, Crecimiento, Maturidad y Declive. Cada fase se divide en las siguientes subfases: Planificación, Análisis, Diseño, Implementación y Prueba. Sus principales características son:

- Se basa en un ciclo de vida en espiral.
- Aborda el riesgo de manera sistemática.
- Proporciona un marco de referencia para el proyecto.
- Permite la integración de las actividades de desarrollo, prueba y validación.
- Pone énfasis en la comunicación y el control.

Metodología de Prototipo:

Es una forma de desarrollar software en la que se crea un prototipo del software antes de comenzar el desarrollo completo. Esto permite que los desarrolladores obtengan una mejor comprensión de lo que el software debe hacer y cómo debe funcionar, lo que a su vez puede ayudar a reducir el tiempo y el costo del desarrollo.

Desarrollo Rápido de Aplicaciones (RAD):

Es un enfoque de desarrollo de software que se centra en la producción de un prototipo funcional lo más rápido posible. El objetivo de RAD es reducir el tiempo de desarrollo acelerando las etapas de análisis, diseño, codificación, prueba e implementación.

Metodología de Programación Extrema (XP):

Es un conjunto de prácticas de desarrollo de software diseñadas para producir software de alta calidad de manera eficiente y en un entorno de cambio constante. XP se centra en la entrega

de software funcional a los clientes a través de un ciclo de desarrollo iterativo e incremental. Las prácticas clave de XP incluyen el diseño extremo, la programación en parejas, la integración continua, la planificación extremadamente corta, las pruebas extremas y la atención extrema a la satisfacción del cliente.

I. Conceptos Fundamentales del Ciclo de Vida del Software:

a. Definición y Importancia

b. Principales Fases del Ciclo de Vida

Imagen 1. Ciclo de desarrollo de software

Argumento: Comprender los conceptos fundamentales del ciclo de vida del software es esencial para garantizar el éxito en el desarrollo de cualquier proyecto. La identificación clara de las fases y su interconexión establecen la base para un proceso de desarrollo eficiente y bien gestionado.

Ejemplo: La implementación del modelo en espiral en el desarrollo de un sistema de gestión empresarial, donde la retroalimentación continua y las iteraciones permitieron ajustar y mejorar constantemente el producto (Boehm, 1988).

II. Metodologías Tradicionales y Ágiles:

a. Modelos en Cascada

b. Desarrollo Iterativo c. Scrum y Kanban

Argumento: La elección entre metodologías tradicionales y ágiles impacta directamente en la velocidad, flexibilidad y calidad del desarrollo. Analizaremos cómo estas enfoques se adaptan a diferentes contextos y proyectos.

Estudio de Caso: El desarrollo de un sistema de comercio electrónico utilizando Scrum, destacando la adaptabilidad a cambios en los requisitos del cliente durante el proceso (Schwaber & Sutherland, 2017).

III. Herramientas y Técnicas del Ciclo de Vida del Software:

a. Control de Versiones

b. Pruebas Automatizadas

c. Integración Continua

Argumento: La implementación efectiva de herramientas y técnicas es esencial para mantener la calidad del software a lo largo de su ciclo de vida. Examinaremos cómo estas prácticas contribuyen a la eficiencia y confiabilidad del producto final.

Ejercicio Práctico: Demostración de la integración continua en un proyecto de desarrollo, resaltando la detección temprana de errores y la mejora del tiempo de entrega (Fowler, 2006).

IV. Evolución y Mantenimiento del Software:

a. Actualizaciones y Parches

b. Refactorización

c. Retiro del Software

Argumento: La fase de evolución y mantenimiento es a menudo subestimada pero crucial para la sostenibilidad a largo plazo. Exploraremos estrategias efectivas para mantener y mejorar continuamente el software después de su implementación.

Vivencia: Experiencia personal en el mantenimiento de un sistema de gestión de contenido, destacando la importancia de actualizaciones regulares y la resolución oportuna de problemas.

Ejemplos Prácticos

1. Modelo en Cascada

Ejemplo de Ciclo de Vida: Sistema de Gestión de Contenidos (CMS) En el modelo en cascada, cada fase del ciclo de vida se realiza secuencialmente. En el desarrollo de un CMS, la fase de diseño se enfocaría en definir la estructura de la base de datos y la interfaz de usuario antes de pasar a la implementación y pruebas.

2. Desarrollo Incremental

Ejemplo de Ciclo de Vida: Aplicación de Comercio Electrónico En un enfoque incremental, podríamos desarrollar una aplicación de comercio electrónico. La primera versión podría permitir la navegación de productos, la siguiente agregar la funcionalidad de carrito de compras y así sucesivamente, entregando incrementos de funcionalidades en cada iteración.

3. Ingeniería de Software Orientado a la Reutilización

Caso de Estudio: Sistema Operativo Android En el ciclo de vida de Android, se observa la ingeniería de software orientado a la reutilización. El núcleo del sistema operativo se reutiliza en una variedad de dispositivos, mientras que las capas superiores pueden adaptarse para satisfacer las necesidades específicas de diferentes fabricantes.

4. Otros Paradigmas

Estudio de Caso: Desarrollo de un Sistema de Control de Versiones en Equipo En un proyecto que requiere una rápida adaptabilidad a cambios, un enfoque ágil podría ser esencial. En el desarrollo de un sistema de control de versiones en equipo, las iteraciones frecuentes permitirían ajustes continuos según las necesidades del usuario.

Conclusión:

Hemos explorado desde su concepción hasta su evolución constante en este recorrido a través del ciclo de vida del software. Hemos demostrado cómo diversas metodologías y prácticas contribuyen al éxito de los proyectos de desarrollo de software a través de estudios de caso, ejemplos prácticos y experiencias reales. Esperamos proporcionar una guía útil para aquellos que buscan comprender y aplicar el ciclo de vida del software en sus proyectos al abordar este tema con un enfoque claro y coherente.

Bibliografía:

- *Ciclo de vida del software: todo lo que necesitas saber*. (s/f). Intelequia. Recuperado el 17 de diciembre de 2023, de https://intelequia.com/es/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber
- *Ciclo Vida Software*. (s/f). Mural.uv.es. Recuperado el 17 de diciembre de 2023, de https://mural.uv.es/givaro/modulo/Ciclo.htm
- (S/f). Amazon.com. Recuperado el 17 de diciembre de 2023, de https://aws.amazon.com/es/what-is/sdlc/
- Boehm, B. W. (1988). "A Spiral Model of Software Development and Enhancement." ACM SIGSOFT Software Engineering Notes, 11(4), 14-24.

- Pressman, R. S. (2014). "Software Engineering: A Practitioner's Approach." McGraw-Hill Education.
- Sommerville, I. (2011). "Software Engineering." Addison-Wesley.
- Schwaber, K., & Sutherland, J. (2017). "The Scrum Guide." Scrum.org. [En línea] Disponible en: https://scrumguides.org/scrum-guide.html
- Fowler, M. (2006). "Continuous Integration." Martin Fowler's Blog. [En línea] Disponible en: https://martinfowler.com/articles/continuousIntegration.html
- Amazon. (2023). Amazon.com. https://aws.amazon.com/es/what-is/sdlc/
- Solera, S. (2022, abril 27). *Las mejores metodologías para un correcto desarrollo de software*. Occamagenciadigital.com. https://www.occamagenciadigital.com/blog/las-mejores-metodologias-para-un-correcto-desarrollo-de-software