EXERCISES QUE TRABAJAREMOS EN LA CUE:

- EXERCISE 1: COMPRENDIENDO SDLC
- EXERCISE 2: MODELOS O METODOLOGIAS DEL SDLC

EXERCISE 1: COMPRENDIENDO SDLC

El "ciclo de vida del desarrollo de software", conocido por sus siglas SDLC en inglés (*software development life cycle*) es un concepto clave en el proceso del desarrollo ya que reduce el costo del desarrollo de software, al mismo tiempo que mejora la calidad y acorta el tiempo de producción. SDLC proporciona un flujo de fases bien estructurado que ayuda a una organización a producir rápidamente software de alta calidad que está bien probado y listo para su uso en producción.

SDLC, logra cumplir con estos objetivos siguiendo un plan que elimina las trampas típicas de los proyectos de desarrollo de software. Ese plan comienza evaluando los sistemas existentes en busca de deficiencias y después define los requisitos del nuevo sistema. Luego crea el software a través de etapas de **análisis**, **planificación**, **diseño**, **desarrollo**, **prueba** e **implementación**. Al anticipar errores costosos, como no pedir retroalimentación al usuario final o al cliente, SLDC puede eliminar el tener que hacer trabajos redundantes y correcciones posteriores.

También es importante saber que hay un fuerte enfoque en la fase de prueba. Como el SDLC es una metodología repetitiva, debe garantizar la calidad del código en cada fase.

Muchas organizaciones tienden a dedicar pocos esfuerzos a las pruebas, mientras que un mayor enfoque en las pruebas puede ahorrarles mucho trabajo, tiempo y dinero.

ETAPAS DEL SDLC

Seguir las etapas del SDLC garantiza que el proceso funcione de manera fluida, eficiente y productiva. Se pueden definir las siguientes etapas en el ciclo de vida.

1. IDENTIFICAR LOS PROBLEMAS ACTUALES

"¿Cuáles son los problemas actuales?" Esta etapa del SDLC obtiene aportes de todas las partes interesadas, incluidos los *clientes*, *vendedores*, *expertos de la industria* y *programadores*. El objetivo de esta fase es entender las fortalezas y debilidades del sistema actual con la meta de mejorarlo.

2. PLANIFICAR

"¿Qué queremos?" En esta etapa, el equipo determina el costo y los recursos necesarios para implementar los requisitos analizados. También detalla los riesgos involucrados y proporciona planes alternativos para minimizar los riesgos. En otras palabras, el equipo debe determinar la viabilidad del proyecto y cómo pueden implementar el proyecto con éxito teniendo en cuenta el riesgo más bajo.

3. DISEÑO

"¿Cómo obtendremos lo que queremos?" Esta fase del SDLC comienza convirtiendo las especificaciones del software en un plan de diseño llamado "Especificación de Diseño". Luego, todas las partes interesadas revisan este plan y ofrecen comentarios y sugerencias. Es fundamental contar con un plan para recopilar e incorporar los aportes de las partes interesadas en este documento.

El fracaso en esta etapa resultará casi con certeza en sobrecostos en el mejor de los casos y en el peor de los casos, el colapso total del proyecto.

4. CONSTRUIR

"A construir lo que queremos". En esta etapa, comienza el desarrollo real. Es importante que cada desarrollador se apegue al plan acordado. Además, aquí también hay que definir las pautas adecuadas sobre el estilo y las prácticas del código a implementar.

Por ejemplo, quizás se defina usar una nomenclatura para archivos o un estilo de nomenclatura de variables como camelCase (una convención tipográfica en la que se usa una mayúscula inicial para la primera letra de una palabra que forma el segundo elemento de una palabra compuesta). Esto ayudará a

tu equipo a producir un código organizado y coherente que sea no solo más fácil de entender sino también más fácil de probar durante la siguiente fase.

5. PRUEBA DE CÓDIGO

"¿Conseguimos lo que queríamos?". En esta etapa, se hacen pruebas de errores y deficiencias. Arreglamos esos problemas hasta que el producto cumpla con las especificaciones originales o, en otras palabras, queremos verificar si el código cumple con los requisitos definidos.

6. IMPLEMENTACIÓN DE SOFTWARE

"Empecemos a usar lo que tenemos". En esta etapa, el objetivo es implementar el software en el entorno de producción, *en donde normalmente se usaría la aplicación*, para que los usuarios puedan comenzar a utilizar el producto. Sin embargo, muchas organizaciones eligen mover el producto a través de diferentes entornos de implementación, como un entorno de prueba. Esto permite que cualquier parte interesada utilice el producto con seguridad antes de lanzarlo al mercado. Además, esto permite detectar cualquier error final antes de lanzarlo.

Como podemos ver, el ciclo de vida es un proceso importante con pasos bien definidos. En el siguiente ejercicio conoceremos más a fondo los diferentes procesos o "modelos" utilizados para cumplir con estas fases generales del SDLC.

EXERCISE 2: MODELOS O METODOLOGIAS DEL SDLC

Ahora que tenemos una mejor comprensión de lo que es el SDLC, podemos comenzar a analizar las diferentes metodologías o modelos utilizados para lograr sus objetivos.

Existen dos enfoques clave para el flujo de trabajo en los SDLC: *el lineal*, basado en un plan tradicional, frecuentemente denominado **cascada**, y el *no lineal*, incremental, frecuentemente denominado **ágil**. Sería una simplificación argumentar que un enfoque es mejor o peor. Es por eso que analizaremos en detalle algunos de los modelos más populares al momento de desarrollar software.

WATERFALL O MODELO DE CASCADA

Este modelo SDLC es el más antiguo y sencillo. Con esta metodología, terminamos una fase y luego comenzamos la siguiente. Cada fase tiene su propio mini-plan y cada fase "cae en cascada" hacía la siguiente.

El mayor inconveniente de este modelo es que los pequeños detalles que se dejan incompletos pueden retrasar todo el proceso.

El método de cascada es una metodología de desarrollo de software impulsada por un plan basado en la planificación avanzada de todas las fases del proyecto, que se implementan posteriormente. Pasar de una fase a otra requiere una verificación y aprobación formal por parte del cliente del software. Este enfoque del desarrollo de software es útil para coordinar el trabajo de un gran proyecto entre diferentes equipos. El inconveniente de la cascada es que es difícil acomodar los cambios necesarios, las correcciones y/o ajustes posibles y necesarios a medida que se desarrollan el proyecto y sus fases. El método de cascada es un método adecuado en todos aquellos casos (raros) en los que todos los requisitos funcionales y no funcionales son claros, bien entendidos y predecibles.

FEATURE DRIVEN DEVELOPMENT (FDD)

El desarrollo basado en características (FDD), es un proceso de desarrollo de software iterativo e incremental. Este proceso consta de 5 fases, la primera es "desarrollar un modelo general". En colaboración con expertos y desarrolladores, se crea un modelo general que captura las funciones clave y sus relaciones en el sistema que debe crearse. Luego, la siguiente fase es “construir una lista de características” en donde cada característica describe una parte del modelo. Una "característica" se define como una pieza pequeña, entregable y valorada por el cliente de la funcionalidad de un sistema, que se puede implementar en no más de 2 semanas. La siguiente fase es la “planificación por característica”, en la que cada característica se analiza cuidadosamente para que cumpla con los requisitos del cliente. Por último, se “diseñan” y “construyen” las funciones, lo que marca la fase final.

Esta metodología es adecuada para proyectos a largo plazo que cambian y agregan funciones continuamente en iteraciones regulares y predecibles. También es útil si un proyecto se vuelve demasiado grande y complejo para que los equipos más pequeños manejen efectivamente la cantidad de trabajo.

AGILE O ÁGIL

El modelo Agile separa el “producto”, o software, en ciclos y entrega un producto funcional muy rápidamente. Esta metodología produce una sucesión de “lanzamientos”, o entregas del software, para ponerlo a prueba. La prueba de cada versión retroalimenta la información que se incorpora en la próxima versión.

El inconveniente de este modelo es que el gran énfasis en la interacción con el cliente puede llevar el proyecto en la dirección equivocada en algunos casos.

Si vamos más al grano, este método se basa en una entrega incremental de funciones de software, es decir, el software se construye paso a paso, por lo que, en cada una de las etapas subsiguientes de la implementación del proyecto, el resultado del proceso se evalúa contra la retroalimentación recibida también del cliente. En otras palabras, a diferencia de la cascada, la retroalimentación se alimenta al proceso de manera instantánea e incremental en lugar de solo al final del proceso. Como resultado, existe la flexibilidad de ir y venir para abordar posibles errores, imprecisiones, o cualquier otro detalle en el software en desarrollo.

El método ágil requiere un trabajo en equipo avanzado y habilidades relacionadas. El flujo de trabajo se divide en los llamados “sprints”, es decir, unidades de trabajo de tiempo limitado durante las cuales se debe entregar un subcomponente específico del proyecto. Los entregables se priorizan de acuerdo con su valor comercial, de acuerdo con las especificaciones y necesidades del cliente. Si no se puede completar todo el trabajo planificado para un sprint determinado, se vuelve a priorizar el trabajo y la información se utiliza para la planificación de sprints futuros.

Esta metodología es muy útil cuando se trata de construir algo innovador que no existe de ninguna forma en la actualidad, ya que permite al propietario del producto descubrir las características y los requisitos del proyecto de forma iterativa.

SCRUM

Scrum es el modelo ágil más popular utilizado en la industria. Este modelo tiene el objetivo de aumentar la eficiencia del proceso de desarrollo de software, incluida la velocidad de entrega y la calidad del software

en sí. Esta metodología es tan popular que es incluso utilizado como una forma de gestión de trabajo en otros dominios distintos del desarrollo de software. En otras palabras, su uso no se limita a la gestión de proyectos de software.

Según scrumguides.org, Scrum es un “modelo ligero y fácil de entender que se utiliza para desarrollar productos complejos”. El scrum es un proceso que se basa en tres pilares: la transparencia, la inspección y la adaptación. Como la metodología ágil, scrum entrega el proyecto en iteraciones de intervalos fijos llamados “sprints”.

Como mencionamos antes, scrum se puede usar incluso en campos que no están relacionados con la programación o el desarrollo de software. Por lo tanto, esta metodología puede ser utilizada simplemente por cualquier persona que necesite crear un producto final.

DEVOPS

Según el sitio de servicios web de Amazon (AWS), "DevOps es la combinación de filosofías, prácticas y herramientas culturales que aumentan la capacidad de una organización para entregar aplicaciones y servicios a alta velocidad: evolucionar y mejorar los productos a un ritmo más rápido que las organizaciones que utilizan procesos de gestión de desarrollo de software tradicionales. Esta velocidad permite a las organizaciones servir mejor a sus clientes y competir de manera más efectiva en el mercado".

Bajo un modelo DevOps, los equipos de desarrollo y operaciones no trabajan de manera independiente o aislada. A veces, estos dos equipos se fusionan en un solo equipo donde los integrantes trabajan en todo el ciclo de vida de la aplicación, desde el desarrollo y las pruebas hasta la implementación y las operaciones, y desarrollan una variedad de habilidades que no se limitan a una sola función. Esta metodología, se utiliza especialmente para garantizar una implementación más rápida del software siguiendo los mismos casos de uso que los modelos agile y scrum.

OTROS MODELOS

Aunque hemos visto en profundidad las metodologías SDLC más comunes, también es bueno conocer algunos otros modelos que también son muy efectivos.

ITERATIVO

Este modelo SDLC enfatiza la repetición. Los desarrolladores crean una versión muy rápidamente y por un costo relativamente bajo, luego la prueban y la mejoran a través de versiones rápidas y sucesivas. Una gran desventaja aquí es que puede consumir recursos rápidamente si no se controla.

BIG BANG

Este modelo de alto riesgo dedica la mayor parte de sus recursos al desarrollo y funciona mejor para proyectos pequeños. Carece de la etapa de definición de requisitos exhaustiva de los otros métodos.

MODELO ESPIRAL

El más flexible de los modelos SDLC, el modelo espiral es similar al modelo iterativo en su énfasis en la repetición. El modelo en espiral pasa por las fases de planificación, diseño, construcción y prueba una y otra vez, con mejoras graduales en cada paso.

Como hemos visto, todos estos métodos nos permitirán pensar más como desarrolladores y comprender el proceso involucrado en equipos de desarrollo altamente eficientes.