Material descargable

Introducción a Quality Assurance

Ciclo de Vida del Desarrollo de Software (SDLC)

Te acercamos de manera independiente la teoría sobre las fases del ciclo de vida del **desarrollo de software** para que puedas repasar sus características

Recuerda, que el desarrollo de software es un proceso sistemático empleado para construir software de manera eficiente y efectiva. Este ciclo abarca diversas fases, cada una con objetivos y actividades específicas. A continuación, detallaremos cada fase del SDLC:

1. **Estrategia / Planificación y Análisis de Requisitos**
    - Objetivo: Definir el alcance del proyecto.
    - Actividades:
        - Identificar las necesidades del cliente.
        - Realizar un análisis de factibilidad.
        - Establecer objetivos claros y definir los requisitos del sistema.
        - Definir el enfoque estratégico para alcanzar los objetivos del proyecto.
        - Establecer un plan detallado que incluye asignación de recursos y plazos.
        - Analizar los riesgos potenciales y propone estrategias de mitigación.
    - Resultado: Documento de requisitos del software.
2. **Diseño del Sistema o Software**
    - Objetivo: Planificar la arquitectura del software.
    - Actividades:
        - Diseñar la arquitectura del sistema.
        - Crear modelos y prototipos.
        - Definir las especificaciones técnicas y los estándares de codificación.

        - Enfocar la planificación arquitectónica para garantizar escalabilidad y flexibilidad.
        - Considerar la seguridad del sistema al definir políticas y controles.
        - Implementar prácticas de diseño centrado en el usuario para mejorar la experiencia del usuario.
    - Resultado: Documento de diseño del software.
3. **Implementación o Codificación / Desarrollo de Software**
    - Objetivo: Construir el software.
    - Actividades:
        - Codificar el software utilizando lenguajes de programación adecuados.
        - Seguir las directrices y estándares definidos en la fase de diseño.
        - Utilizar prácticas de codificación segura para prevenir vulnerabilidades.
        - Facilitar la colaboración entre los desarrolladores a través de sistemas de control de versiones.
        - Documentar el código para facilitar futuras actualizaciones y mantenimiento.
    - Resultado: Software en su versión inicial.
4. **Pruebas e integración**
    - Objetivo: Asegurar que el software funcione correctamente.
    - Actividades:
        - Realizar pruebas unitarias, de integración, de sistema y de aceptación.
        - Identificar y corregir errores y defectos.
        - Implementar pruebas de rendimiento para evaluar la capacidad del software bajo carga.
        - Integrar herramientas de automatización de pruebas para mejorar la eficiencia.
        - Realizar pruebas de seguridad para identificar posibles vulnerabilidades.
    - Resultado: Software probado y listo para la implementación.
5. **Implementación o Despliegue**

- Objetivo: Desplegar el software en el entorno del usuario.
- Actividades:
  - Instalar y configurar el software.
  - Realizar migraciones de datos si es necesario.
  - Capacitar a los usuarios finales.
  - Realizar una evaluación previa al lanzamiento para garantizar la compatibilidad con el entorno de usuario.
  - Proporcionar manuales y recursos de capacitación para facilitar la adopción del software.
  - Establecer un plan de contingencia para abordar posibles problemas post-implementación.
- Resultado: Software en funcionamiento en el entorno del cliente.

6. **Mantenimiento y Soporte / Operacionalización y Mantenimiento**
   - Objetivo: Asegurar el funcionamiento continuo y eficiente del software.
   - Actividades:
     - Corregir errores que aparecen después de la implementación.
     - Mejorar el rendimiento del software.
     - Actualizar el software según sea necesario.
     - Implementa monitoreo continuo para detectar y abordar problemas proactivamente.
     - Establecer un proceso de retroalimentación del usuario para recopilar comentarios y realizar mejoras.
     - Evaluar la viabilidad de actualizaciones tecnológicas para mantener el software alineado con las tendencias actuales.
   - Resultado: Software actualizado y en funcionamiento a largo plazo.

Habiendo revisado las distintas fases del SDLC, es evidente que la intervención de un tester es esencial en cada una de ellas. La forma en que el tester actuará dependerá de factores específicos de cada empresa, como los tiempos y recursos disponibles, entre otros.

Para resumir, a continuación se detallan algunas de las formas de intervención en cada fase, conceptos que ampliarás con posterioridad:

- **Fase de Requisitos**

    - Cuándo interviene: Desde el inicio del proyecto, cuando se definen los requisitos del cliente o del mercado para el software.
    - Cómo interviene: El tester participa en la revisión de requisitos para identificar posibles lagunas, ambigüedades o inconsistencias. Contribuye a asegurar que los requisitos sean claros, comprensibles y verificables.
- **Fase de Diseño:**
    - Cuándo interviene: Durante la creación del diseño del software.
    - Cómo interviene: El tester revisa los documentos de diseño para asegurarse de que cumplan con los requisitos establecidos. Identifica posibles problemas en la arquitectura que podrían afectar la calidad del software.
- **Fase de Implementación (Desarrollo):**
    - Cuándo interviene: A medida que se implementa el código.
    - Cómo interviene: Realiza pruebas unitarias para garantizar que cada componente del software funcione correctamente. Identifica y reporta defectos a los desarrolladores para su corrección.
- **Fase de Pruebas Integradas:**
    - Cuándo interviene: Después de la implementación de componentes individuales, se realizan pruebas integradas para evaluar la interacción entre ellos.
    - Cómo interviene: Ejecuta pruebas de integración para identificar posibles problemas de compatibilidad y asegurarse de que los diferentes módulos del software trabajen de manera conjunta.
- **Fase de Pruebas del Sistema:**
    - Cuándo interviene: Antes del lanzamiento del producto al público.
    - Cómo interviene: Realiza pruebas exhaustivas del sistema para garantizar que cumpla con los requisitos del cliente y funcione correctamente en diferentes entornos. Identifica y documenta cualquier defecto que pueda afectar la calidad del producto final.
- **Fase de Mantenimiento:**
    - Cuándo interviene: Después del lanzamiento, durante el ciclo de vida del software.

- Cómo interviene: Continúa realizando pruebas de regresión para asegurar que las actualizaciones o cambios no introduzcan nuevos defectos. Colabora con el equipo de desarrollo para corregir cualquier problema que surja en el uso continuado del software.