UNIDAD II - EVALUACIÓN
EST319 - Ingeniería de Software

Alumno: Maquera Andrade Aldair Jose
Código:214540

Lea cuidadosamente y resuelva

1. Explique el proceso completo de evaluación de software y discuta cómo se pueden aplicar diferentes técnicas de evaluación en las distintas fases del ciclo de vida del software.

   El proceso de evaluación de software incluye las fases de planificación, especificación, diseño, ejecución, análisis y reporte, que aseguran una buena calidad de software(producto).

   En la planificación se establecen objetivos y criterios de calidad, utilizando técnicas como revisión de requisitos. Durante el análisis y diseño, se evalúa la arquitectura y el diseño mediante revisiones y el prototipado. En segundo la implementación, es donde se verifica el código con revisiones con el fin de verificar la funcionalidad con pruebas unitarias, de integración, sistema y aceptación. Luego el mantenimiento incluye evaluación continua con monitoreo al prududcto. Para terminar, la evaluación post-implementación analiza el éxito del proyecto mediante encuestas y métricas. Todo esto con el fun de garantizan que el software cumpla con los requisitos y funcione adecuadamente en cada fase del desarrollo.

2. Describa al menos tres métricas de evaluación de software, explicando cómo se calculan y en qué situaciones específicas cada una de ellas resulta más útil.

   - **Indice de Casos de Uso (UCP):** El método UCP utiliza actores y casos de uso para estimar el tamaño del software. A cada uno se le asigna una ponderación basada en su complejidad.

     La fórmula básica para calcular los puntos de casos de uso es:

     $$UCP = UUCP \times TCF \times ECF$$

     donde: - $UUCP$ es la suma de las ponderaciones de actores no ajustadas (UAW) y las ponderaciones de casos de uso no ajustadas (UUCW). - TCF es el Factor de Complejidad Técnica. - ECF es el Factor de Complejidad Ambiental.

   - **Acoplamiento en el Software** El acoplamiento es una métrica fundamental en la calidad del software, complementaria a la cohesión [1]. Se refiere a cómo están interconectadas las entidades dentro de un sistema de software, lo cual tiene un impacto significativo en la mantenibilidad del producto

El acoplamiento (C) se calcula utilizando la siguiente fórmula:

$$\text{Acoplamiento}(C) = 1 - \frac{1}{di + 2 \times ci + do + 2 \times co + gd + 2 \times gc + w + r}$$

Donde:

- di: número de parámetros de entrada
- ci: número de parámetros de control
- do: número de parámetros de salida
- co: número de parámetros de control de salida
- gd: número de variables globales usadas como datos
- gc: número de variables globales usadas como control
- w: número de módulos llamados (fan-out)
- r: número de módulos llamantes bajo consideración (fan-in)

El valor de *Acoplamiento* (C) aumenta con el nivel de acoplamiento del módulo. Este valor varía aproximadamente entre 0.67 (bajo acoplamiento) y 1.0 (alto acoplamiento).

- **Tiempo Promedio de Localización de Defectos (MTTD)** El Tiempo Promedio de Localización de Defectos (MTTD) se define como el promedio del tiempo transcurrido entre la introducción de un defecto y su detección. Esta métrica es crucial para evaluar la eficiencia en la detección de defectos y mejorar la calidad del software [2].

La fórmula para calcular el MTTD es:

$$\text{MTTD} = \frac{\sum_{i=1}^{n} \text{Tiempo para detectar cada defecto}}{\text{Número total de defectos}} \tag{1}$$

Donde:

- n es el número total de defectos.
- Tiempo para detectar cada defecto es el tiempo transcurrido desde la introducción hasta la detección de cada defecto.

Esta fórmula proporciona una medida cuantitativa del tiempo promedio que toma detectar un defecto en el proceso de desarrollo de software.

3. Analice las diferencias entre la calidad del producto y la calidad del proceso en el contexto del desarrollo de software, proporcionando ejemplos concretos de cómo cada tipo de calidad puede ser medido y mejorado.

   La calidad del producto y la calidad del proceso van de la mano se puede decir son las 2 caras de una moneda, ambas son esenciales para lograr un producto de calidad (software).La calidad del producto se enfoca en el software final, midiendo funcionalidad, usabilidad y eficiencia. Por otro lado, la calidad del proceso se centra en cómo se desarrolla el software, evaluando métricas como tiempo de entrega, tasa de errores y eficiencia del equipo

Un ejemplo de medición de calidad del producto sería evaluar el tiempo de respuesta de la aplicación, mientras que para la calidad del proceso se podría medir el tiempo promedio para resolver errores. Ambos tipos de calidad son complementarios y esenciales para el éxito global del proyecto de software.

4. Suponga que necesita implementar una estrategia de respaldo para una base de datos crítica. Escriba un script en SQL para realizar un respaldo completo de la base de datos, y explique los pasos y consideraciones clave en este proceso

   En el cmd se escribe la siguiente linea donde se cambia el usuario y contraseña; tambien cambiar la base de datos se se dese guardar

   ```
   mysqldump -u root -p951024451 unlockopia >

   C:\Users\Alenm\Desktop\Ecommerce\Backup\unlockopia.sql
   ```

   Para asegurar el backup, este cifrado:

   ```
   gpg -c backup.sql
   ```

   Para automatizar la tarea es colocarlo .bat y guardarlo en programador de tareas

5. Discuta la ISO/IEC 25010, su importancia en la industria del software, y cómo puede ser implementada para asegurar la calidad de un producto software.

   La ISO/IEC 25010 es importante para establecer un marco claro y efectivo para evaluar y mejorar la calidad del software. Su implementación ayuda en la satisfacción del cliente y y eficiencia en el mantenimiento de software.

6. Explique cómo las métricas de evaluación de software pueden influir en la toma de decisiones durante el desarrollo de un proyecto, y describa un caso práctico donde el uso de métricas podría mejorar significativamente la calidad del software entregado.

   Las métricas de evaluación de software influyen durante el desarrollo de proyectos, proporcionando datos objetivos para guiar acciones y mejoras. Estas métricas ofrecen insights sobre la calidad, eficiencia y progreso del proyecto, permitiendo a los equipos identificar áreas problemáticas, priorizar tareas y asignar recursos de manera efectiva. Por ejemplo, métricas de complejidad de código pueden indicar la necesidad de refactorización, mientras que las tasas de defectos pueden señalar la necesidad de más pruebas o revisiones de código.

   Caso práctico: En el desarrollo de una aplicación móvil, el equipo implementa métricas de rendimiento y usabilidad. Al monitorear el tiempo de carga de la aplicación y la tasa de errores de usuario, descubren que ciertas funciones tienen tiempos de respuesta inaceptables y provocan frecuentes errores de usuario. Esta información les permite priorizar la optimización de estas funciones específicas, resultando en una mejora significativa en la experiencia del usuario y la calidad general de la aplicación antes de su lanzamiento.

Referencias

[1] Ioan Gabriel Czibula et al. "An aggregated coupling measure for the analysis of object-oriented software systems". En: *Journal of Systems and Software* (2019). URL: https://www.sciencedirect.com/science/article/abs/pii/S0164121218302371.

[2] Faheem Ullah y Thomas R. Gross. "Profiling for Detecting Performance Anomalies in Concurrent Software". En: (2015), págs. 1-10. URL: https://ethz.ch/content/dam/ethz/special-interest/infk/inst-cs/lst-dam/documents/Publications/SEPS_2015.pdf.