Orlando Ulises Aguilar Rojas
173118

24.1. Explique por qué un proceso de software de alta calidad debería conducir a productos de alta calidad de software. Discuta los posibles problemas con este sistema de gestión de calidad.

Tomando en cuenta los procesos para realizar productos, el configurar, establecer opciones y todo lo que involucra a las maquinas para realizar estos productos de alta calidad, el software también debe de ir más allá del proceso mecánico, siendo de alta calidad debido a mas cosas externas del propio sistema.

24.2. Exponga cómo pueden usarse los estándares para obtener conocimiento de la organización sobre los métodos efectivos de desarrollo de software. Sugiera cuatro tipos de conocimiento que puedan reflejarse en los estándares de la organización.

1. Conocimiento de tipos específicos por defecto que ocurren en el departamento del desarrollo de la organización. Esto entra en los estándares de revisión.
2. Conocimiento por modelo del sistema que mejora la utilidad del mantenimiento del sistema. Esto entra en el diseño estándar de documentación
3. Conocimiento de las herramientas de soporte que puede ser útil en los proyectos. Esto entra en los estándares de los ambientes de desarrollo usados por todos los proyectos.
4. Conocimiento en los tipos de información útil para incluir como comentarios en código. Esto entra en el estándar de comentarios en código.

24.3. Discuta la valoración de calidad del software según los atributos de calidad mostrados en la figura 24.2. Debe considerar a la vez cada atributo y explicar cómo puede valorarse.

Protección	Comprensibilidad	Portabilidad
Seguridad	Comprobabilidad	Usabilidad
Fiabilidad	Adaptabilidad	Reusabilidad
Flexibilidad	Modularidad	Eficiencia
Robustez	Complejidad	Facilidad para que el usuario aprenda a utilizarlo

En general, cada atributo cumple con una función específica dentro del

desarrollo seguro de software, cada una de ellas, tiene una prioridad

diferente a la otra, en algunos casos, 2 o más atributos son prioridad alta en

el desarrollo.

En lo personal, los 3 atributos que tienen más peso y que le pueden dar una

buena presentación al software y que sea reconocido, son:

1. Protección

2. Seguridad

3. Fiabilidad

Orlando Ulises Aguilar Rojas
173118

24.4. Diseñe un formato electrónico que pueda usar para registrar comentarios de revisión y para enviar comentarios por correo electrónico a los revisores.

Un formulario simple que envía metadatos con la fecha de hoy es directamente enviado al departamento del asesor para que pueda ser consultado.

Orlando Ulises Aguilar Rojas
173118

24.5. Describa brevemente posibles estándares que podría utilizar para:

Basándonos en el libro, el responder algunas preguntas y se utilizará para estandarizar lo siguiente

- El uso de sentencias de control en C, C# o Java;

  Metodología de programación

  Estándar para el nombre de las variables

  Cantidad de tabulaciones dentro de sentencias

- enviar reportes para un proyecto final en una universidad;

  Datos de contacto

  Planteamiento de cosas a revisar

  Teoría

  Metodología

  Forma de presentación dependiendo el formato utilizado

- el proceso de hacer y aprobar cambios al programa (véase el capítulo 26);

  Comprobar requerimientos

  Comprobar cambios

  Aplicar testing

  Informar cambios

- el proceso de comprar e instalar una nueva computadora.

  Comprobar requerimientos

  Establecer costos

  Comprar piezas necesarias

  Comprar licencia o descargar sistema operativo

  Armar computadora

  Actualizar BIOS en caso de ser necesario

  Instalar sistema operativo

  Instalar software requerido por el usuario

Orlando Ulises Aguilar Rojas
173118

24.6. Suponga que trabaja en una organización que desarrolla productos de bases de datos para individuos y empresas pequeñas. Esta organización está interesada en cuantificar su desarrollo de software. Escriba un reporte que sugiera métricas adecuadas y mencione cómo pueden recopilarse.

Puede hacerse por el índice de fog, pues es la medida de legibilidad, el numero de fallas y el número de días-hombre requeridas.

asignaríamos valores a los atributos de la calidad del sistema, para medir su complejidad ciclomatica y valorara los atributos de la calidad del sistema.

Puede tomarse en cuenta:

número de sentencias enviadas

Tiempo de CPU usado

Frecuencia de errores.

24.7. Exprese por qué las inspecciones de programa son una técnica efectiva para descubrir errores en un programa. ¿Qué tipos de errores tienen escasa probabilidad de descubrirse mediante inspecciones?

Las inspecciones de programa son una idea antigua y la mayoría de los estudios y experimentos indican que las inspecciones son más efectivas para el descubrimiento de defectos, que para las pruebas del programa. Fagan reportó que más del 60% de los errores en un programa se detectan mediante inspecciones informales de programa. En el proceso de Cleanroom se afirma que más del 90% de los defectos pueden detectarse en inspecciones del programa.

Se tiene la probabilidad baja de encontrar errores lógicos, pues la maquina en si hace lo que esta programado y el que puede fallar es la lógica del programador

24.8. Diga por qué las métricas de diseño son, por sí mismas, un método inadecuado para predecir la calidad del diseño.

Por la falta de metas bien definidas, son inadecuadas debido a las métricas para conseguir estas metas y el nivel de desarrollo de estas, son genéricas y afecta en la calidad del software.

24.9. Exponga por qué es difícil validar las relaciones entre atributos de producto internos (como la complejidad ciclomática) y los atributos externos (como la mantenibilidad).

No habría relación. Se buscaría otra forma de relacionarlos

24.10. Un colega que es muy buen programador elabora software con un bajo número de defectos, pero siempre pasa por alto los estándares de calidad de la organización. ¿Cómo deberían reaccionar sus administradores ante este comportamiento?

Se debería de replantear el pensar que tan bueno es, quizá hace lo que debe, pero es difícil hacer algunos procesos que plantea los estándares es para mejorar a futuro este software. Yo creo que debería tomarse el tiempo para mejorar la calidad de software, pues si es bueno resolviendo los problemas planteados, la capacitación de mejora en la calidad es secundario.