INVESTIGACIÓN REQUISITOS NO FUNCIONALES

POR

ASTRID CAROLINA DÍAZ GÓMEZ

LUIS CARLOS MARIN CAMPOS

ANDRES DARIO HIGUITA PEREZ

LUIS MATEO HINCAPÍE MARTÍNEZ

ARQUITECTURA DE SOFTWARE

INTRUCTOR:

ROBINSON CORONADO GARCIA

INGENIERÍA DE SISTEMAS

UNIVERSIDAD DE ANTIOQUIA

1 8 0 3

UNIVERSIDAD DE ANTIOQUIA

2020-1

Introducción

La importancia de los requisitos no funcionales, toman fuerza a la hora de llevar un producto de calidad a un usuario, logrando su mantenibilidad y escalabilidad.

Objetivo General

Nuestro objetivo general es conocer los conceptos de los requisitos no funcionales, los cuales serán un factor importante en el momento de hacer parte de un equipo de desarrollo de alto rendimiento.

Objetivos Específicos

Como objetivos específicos nos dispondremos a investigar sobre los siguientes requisitos no funcionales:

- Testability
- Reusability
- Reliability
- Integrability
- Legibilidad
- Fault Tolerance
- Open Source

Procedimiento

Integrability

Características:

- Integralidad significa la capacidad de hacer que los componentes desarrollados por separado de un sistema funcionen juntos correctamente.
- La integralidad está relacionada con la interoperabilidad y nuevamente, la interconectividad.
- La integralidad tiene un impacto decisivo en el desarrollo y evolución de un sistema, por lo que debe tenerse en cuenta, así como las demás características de una familia de sistemas, como los requisitos de dominio, los elementos arquitectónicos de grano grueso y las prácticas utilizadas para el desarrollo y desarrollo. mantener una familia de sistemas y obtener productos de ella.

La integrabilidad depende de:

- complejidad de las componentes
- mecanismos y protocolos de comunicación
- claridad en la asignación de responsabilidades
- calidad y completitud de la especificación de las interfaces

Testability

- Facilidad con la cual el software puede mostrar sus defectos (típicamente a través de pruebas de ejecución).
- Probabilidad de que, suponiendo que el software tiene al menos un defecto, fallará en la siguiente prueba.
- Condiciones necesarias:
    - controlabilidad - poder controlar el estado interno de las componentes
    - observabilidad - poder observar las salidas (outputs)

Reusability

- Es la capacidad que tiene un sistema para que su estructura o alguna de sus componentes puedan ser usadas en futuras aplicaciones.
- La reusabilidad se relaciona con la arquitectura en que las componentes son las principales unidades de reutilización.
- la reusabilidad dependerá del acoplamiento de la componente, reusar una componente implica reusar la clausura transitiva de todas las componentes dependientes en la estructura de uso.
- La reusabilidad es una forma particular de modificabilidad

Ejemplo:
Un sistema S está compuesto de 100 componentes (S1 a S100). Se construye otro sistema T, y se descubre que las n primeras componentes son idénticas y pueden reutilizarse.

Legibilidad

La legibilidad del código es conseguir un software que esté a la altura de codificación que permitan tener una deuda técnica aceptable y faciliten la evolución del sistema de información.

Escribiendo el código usando patrones de diseño y con el nombramiento de las clases y variables. Un programa puede hacerse más legible dándole cierto formato al código, utilizando el sangrado (indentación) para reflejar las estructuras de control del programa, e insertando espacios o tabuladores.

Validity

Esta se refiere a la validación que se da a los datos, asegurando la entrega de datos limpios y claros entre programas, aplicaciones y servicios que lo utilizan.

La validación de datos ayuda principalmente a garantizar que los datos enviados a las aplicaciones conectadas sean completos, precisos, seguros y consistentes. Esto se logra a través de controles de validación de datos y reglas que rutinariamente comprueban la validez de los datos. Estas reglas generalmente se definen en un diccionario de datos o se implementan a través de un software de validación de datos.

Algunos de los tipos de validación de datos incluyen:

- Validación de código
- Validación de tipo de datos
- Validación del rango de datos
- Validación de restricciones
- Validación estructurada

Reliability

DEFINICIÓN: La confiabilidad es la medida en que el sistema de software realiza de manera consistente las funciones especificadas sin fallas.
ELICITACIÓN: Los requisitos de confiabilidad abordan la preocupación del usuario por la inmunidad del sistema a fallas. Al obtener requisitos de confiabilidad, considere las necesidades con respecto a las posibles causas de fallas del sistema, las acciones preventivas o los procedimientos necesarios para evitar fallas, clases de fallas y métricas de confiabilidad.

EJEMPLO: [Probabilidad de falla a pedido] La probabilidad de falla a pedido del sitio web de RQ (POFOD) será 0,0001 (1 de cada 10000 reproducciones) cuando un estudiante solicite reproducir un video del curso.

Fault Tolerance

Es el grado en el que un sistema, producto o componente funciona según lo previsto a pesar de la presencia de fallas de hardware o software.
Un diseño tolerante a fallas permite que un sistema continúe con la operación prevista, posiblemente a un nivel reducido, en lugar de fallar por completo, cuando falla alguna parte del sistema.
La tolerancia a fallas se puede lograr anticipando condiciones excepcionales y construyendo el sistema para enfrentarlas y, en general, apuntando a la autoestabilización para que el sistema converja hacia un estado libre de errores.
Por ejemplo, un edificio con un generador eléctrico de respaldo proporcionará el mismo voltaje a los enchufes de pared incluso si falla la energía de la red.
Los sistemas tolerantes a fallas se basan generalmente en el concepto de redundancia.
Redundancia es la provisión de capacidades funcionales que serían innecesarias en un entorno libre de fallas. Respaldo.
Por ejemplo, las tractomulas tienen muchas llantas, y ninguna es crítica (Excepto las delanteras porque no soportan tanto peso).

Open Source

El concepto de código abierto se centra en la suposición de que al permitir la visualización y modificación del código, los usuarios desarrollaran un software de calidad superior al software propietario.

Según lo establecido por la Open Source Iniciative, el software de código abierto debe reunir una serie de criterios para ser considerado como tal. Dichos requisitos son:

- **Libre redistribución**: el software debe poder ser regalado o distribuido libremente.
- **Código fuente**: el código fuente debe estar incluido u obtenerse libremente.
- **Trabajos derivados**: la redistribución de modificaciones debe estar permitida.
- **Integridad del código fuente del autor**: las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
- **Sin discriminación** de personas o grupos.
- **Distribución de la licencia**: deben aplicarse los mismos derechos a todo el que reciba el programa.
- **La licencia no debe ser específica de un producto**: el programa no puede licenciarse sólo como parte de una distribución mayor.

- **La licencia no debe restringir otro software**: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
- **La licencia debe ser tecnológicamente neutral**: no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

Webgrafía

https://jummp.wordpress.com/2011/08/27/desarrollo-de-software-la-legibilidad-del-codigo/

https://www.ticportal.es/glosario-tic/open-source-codigo-abierto

https://www.tecnologias-informacion.com/validacion.html