Requisitos No Funcionales

UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

Por:

Anderson Barrientos Parra
C.c: 1017242181
Stiven Guerra Chaverra
C.c: 1037672655
Sebastián Gómez Ramírez
C.c: 1045026756
Hermen Esteban Theran Martínez
C.c: 1035875072

Robinson Coronado García
Profesor

Arquitectura de Software - Ingeniería de Sistemas
Universidad De Antioquia
Medellín 2020

Introducción

El presente informe expone, describe y explica algunos de los requisitos no funcionales que debe tener un sistema para alcanzar un buen estándar de calidad, además este informe también presenta y define algunas prácticas que son utilizadas para aumentar la productividad y eficiencia en la creación de software.

Es importante conocer y poner en práctica cada uno de estos requisitos a la hora de desarrollar un software ya que son indicadores que mejoran sustancialmente la eficiencia del producto final.

Objetivo General

Determinar qué propiedades o restricciones son necesarias a la hora de aplicar a un sistema para alcanzar estándares altos de calidad.

Objetivos Específicos

- Dar a conocer algunos de los requisitos no funcionales que necesita un sistema y la importancia de su aplicación.
- Definir en qué etapa del desarrollo del sistema se pueden implementar los requisitos no funcionales.
- Aplicar los requisitos no funcionales en la etapa que se haya definido.

Requisitos no funcionales

Son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o Estándares de Calidad. Son las propiedades o cualidades que el producto debe tener.
Los requisitos no funcionales se originan en la necesidad del usuario, debido a restricciones presupuestarias, políticas organizacionales, la necesidad de interoperabilidad con otros sistemas de software o hardware, o factores externos tales como regulaciones de seguridad, políticas de privacidad, entre otros.

● Continuidad

El requisito de continuidad consta de permitir que un sistema proveedor de servicios, siempre esté disponible para los usuarios aunque resulten

problemas de causa mayor que puedan impedir el correcto funcionamiento de estos. Con este requisito se busca que el sistema esté preparado para afrontar cualquier tipo de interrupción física en el hardware que lo aloja.

Algunas de estas interrupciones pueden ser inundaciones, cortes de energía, entre otros. Es decir, estas interrupciones son dadas por factores que no es posible controlar por software e incluso por hardware.

Etapa de diseño a la que corresponde
Este requisito corresponde a la etapa de desarrollo de *diseño*, donde se busca planificar y establecer la infraestructura que se usará para alojar e implementar el sistema en desarrollo.

Ejemplo y evidencia de aplicación
Un ejemplo de la implementación de este requisito no funcional, es el uso de varios servidores que pueden funcionar en centros de operaciones distribuidos geográficamente, ya que así si un equipo servidor falla, estarán los otros para contrarrestar la demanda de uso de los usuarios.

Otro ejemplo es el control de errores internamente entre los equipos que contienen los servidores, siendo estos mensajes de error y las acciones de corrección manejados internamente sin que el usuario final se de cuenta de lo que sucede al consumir los servicios ofrecidos.

- ## Coexistencia

Se refiere a la capacidad que tiene un producto software para co-existir con otro producto software de manera independiente dentro de un mismo entorno de trabajo, donde ambos software compartirán de manera eficiente los recursos provistos por el sistema. Esto nos permite ahorrar costos en cuanto a hardware se refiere, ya que un mismo equipo nos sirve para alojar todos los software que se requieren para el correcto funcionamiento del sistema.

Etapa de diseño a la que corresponde
La coexistencia pertenece a la etapa de *análisis* en el proceso de desarrollo del proyecto, donde se analizan los recursos a consumir y la eficiencia al compartir estos con otros software.

Ejemplo y evidencia de aplicación
Un ejemplo de coexistencia es: Poderse instalar en una sola computadora todas las aplicaciones, ya sean tanto de servicios web como aplicaciones web, pero siendo estos softwares independientes.

- ## Cambiabilidad

Es un requisito que describe la capacidad o necesidad de un producto software para ser modificado, estas modificaciones pueden implementar algunas correcciones ya sea de un defecto, hacer un determinado cambio en el software al usar nuevas tecnologías o al ser actualizadas las necesidades del modelo de negocio.

Etapa de diseño a la que corresponde
Este requisito pertenece a la etapa de *análisis* en el proceso de desarrollo del proyecto, donde se busca determinar los puntos claves que permitirán incluir los nuevos cambios sin afectar el desarrollo o funcionamiento del proyecto.

Ejemplo y evidencia de aplicación
Un ejemplo de cambiabilidad es cuando una aplicación móvil tiene módulos preparados para añadir nuevas funciones, de forma que sólo es necesario integrar las librerías nuevas y adecuar visualmente la nueva funcionalidad, cabe mencionar que esta nueva funcionalidad no interfiere en el funcionamiento de las ya existentes.

- ## Interoperability

La interoperabilidad se refiere a la facilidad que posee un sistema para comunicarse y compartir información con aplicaciones y subsistemas internos, haciendo que sea incluso más fácil interactuar con sistemas externos.

La interoperabilidad se debe considerar como un requisito no funcional que puede actuar en diferentes áreas de los sistemas:

- Como por ejemplo el hardware, donde se debe hacer la consideración de los dispositivos en los que debe operar el software y que este pueda comunicarse a través de los distintos dispositivos.

- También se puede ver cómo la interoperabilidad de información, donde se evalúa cómo se debe manipular y generar la información para que pueda interactuar y ser compatible con la información generada o manipulada por otra sección del sistema.

- Finalmente, para el caso, se presenta la interoperabilidad técnica, la cual se define a un nivel físico de desarrollo, donde se evalúa la compatibilidad que tienen las diferentes aplicaciones entre sí y su facilidad para tener interacción por medio de un servicio que usan en común. Siendo este un aspecto muy importante ya que se suele trabajar en equipos de desarrollo con enfoques específicos, sin preocuparse mucho por la compatibilidad que tendrán las otras aplicaciones con el mismo servidor.

Etapa de diseño a la que corresponde

Este requisito no funcional corresponde a las etapas de *construcción* ya que son aspectos que se deben considerar a la hora de diseñar el software y van a generar problemas principalmente en la etapa de desarrollo y de no plantear bien las situaciones necesarias, se debe limitar el alcance de la aplicación por no poder cumplir con interoperabilidad entre los sistemas relevantes.

Ejemplo y evidencia de aplicación

Un ejemplo bastante entendible donde se puede comprender el problema de no aplicar correctamente este requisito no funcional es a la hora de usar una aplicación bancaria, al utilizar la aplicación móvil a través de un teléfono inteligente, se realiza una transacción a través de un código QR y la transacción es efectuada exitosamente y queda registrada en el historial de movimientos. Pero al utilizar la aplicación online e intentar ver el historial de movimientos, no aparece esta última transacción mencionada, debido a que la transacción por QR solo se puede realizar a través de la aplicación móvil y el equipo de la aplicación online nunca consideró desplegar esta información. Siendo evidente un problema de interoperabilidad tanto a nivel lógico básico, donde no se realizó una compatibilidad de ambos softwares, como a nivel técnico donde los equipos no consideraron la interacción entre ambas aplicaciones.

- ## Response Time

El tiempo de respuesta está ligado al desempeño, el cual es aquel factor que define qué tan rápido una aplicación de software responde a determinadas acciones realizadas por el usuario, o en otras palabras: qué tanto debe esperar el usuario antes de que la acción deseada suceda.

Jakob Nielsen (una de las personas más respetadas en el ámbito mundial sobre la usabilidad web) habla sobre 3 límites de tiempo de respuesta, los cuales están determinados por las habilidades de percepción humana:

1. **0.1 segundo:** En este margen de tiempo el usuario percibe un tiempo de respuesta como inmediato.
2. **1 segundo:** Si bien el flujo no se ve afectado, el usuario notará un poco de lentitud en la realización de la tarea.
3. **10 segundos:** Se habrá perdido toda la atención del usuario, ya que en este margen de tiempo el usuario preferirá realizar otras actividades mientras espera a que se termine la operación. En la mayoría de los casos, los usuarios abandonarán la página que están visitando.

Estas métricas de las cuales habló en su momento Jakob Nielsen, demuestran lo importante que es el tiempo de respuesta y lo rápido que los usuarios pueden perder el interés por la actividad que están intentando realizar y bien dice, que incluso unos pocos segundos de retraso crean una experiencia de usuario poco placentera.
Cuando se habla de tiempo de respuesta, hay otros factores que se deben tener en cuenta, como lo son la concurrencia y la carga; ya que según el tipo de software o tarea a la que nos refiramos, estos deben ser capaces de responder adecuada y velozmente a sus tareas, ya sea por ejemplo, un servicio REST, consultas a una base de datos, procesos en una aplicación, etc.

Etapa de diseño a la que corresponde
Este requisito no funcional corresponde a las etapas de *operación* y al ser uno de los que son percibidos más fácilmente por el usuario, lo ideal sería un tiempo de respuesta adecuado para asegurar una experiencia de usuario placentera.

Ejemplo y evidencia de aplicación
Hay varios ejemplos en los que este requisito no funcional se aplica de manera adecuada y en los que por lo contrario, no lo hace. Un primer ejemplo, se puede ver en el ámbito de los videojuegos, en el que se espera que cada acción que realice el usuario tenga una reacción inmediata: si el usuario (jugador en este caso) desea que su personaje salte, este debe ver

que en efecto el personaje realice la acción tan pronto como oprime un botón y un retraso de pocos segundos puede arruinar completamente su experiencia.

Otro ejemplos se pueden ver al navegar por una página web o hacer uso de una aplicación, en los cuales el usuario espera que acciones como clicks, transiciones, transacciones tengan una respuesta inmediata, ya que al usuario no le importa (y no tiene porqué importarle) la razón por la cual el tiempo de respuesta es lento, simplemente lo reconocerá como algo molesto.

- ## Manageability

La manejabilidad se define como la facilidad que presenta un sistema para llevar a cabo actividades de administración a través de herramientas diseñadas precisamente para realizar operaciones internas en la aplicación de tal manera que no se deba modificar a nivel técnico.
Es la habilidad de un sistema de proveer información sobre él mismo al equipo encargado del monitoreo, lo cual proporciona ciertas ventajas, como depurar, analizar y encontrar la causa de fallas.

Etapa de diseño a la que corresponde
Este requisito no funcional corresponde a la etapa de *operación* ya que sus características principales se basan en brindarle al encargado de la administración de la aplicación, herramientas que faciliten el monitoreo del rendimiento de la misma como también proporcionar acceso directo a la información manipulada a través del sistema.

Ejemplo y evidencia de aplicación
Un ejemplo básico de una aplicación en la cual se aplica correctamente el requisito no funcional de Manejabilidad es fácilmente evidenciable en el sistema de anuncio de eventos de un videojuego, el cual está vinculado con una aplicación web extra la cual permite monitorear y administrar fácilmente los distintos eventos que tienen lugar en el videojuego, permitiendo controlar aspectos como el tiempo de inicio de un evento, la recompensa, los requisitos de los jugadores para participar y la duración o tiempo de finalización del evento. Finalmente, presentando un cumplimiento bastante acertado del requisito no funcional de Manejabilidad.

- ## Capacity

El requisito no funcional de la Capacidad se define como las posibilidades que tiene un sistema para mejorar su rendimiento a través del aumento de potencia de hardware, todo en pro de ofrecer suficiente funcionalidad cuando se requiera. Dado el caso de que en un sistema se hagan 5000 peticiones por segundo y el sistema solo pueda soportar hasta 2000 peticiones a la vez, se generará un problema de Capacidad siempre y cuando este problema sea generado solo por las limitaciones de hardware. Pese a que este último caso pueda parecer un problema de Disponibilidad, hay que hacer la distinción de que no lo es, ya que el problema presentado no demuestra que haya un error interno que impida el uso de las 2000 peticiones al tiempo, sino que simplemente no se pueden ejecutar más de la cantidad de peticiones definidas.

Etapa de diseño a la que corresponde

Este requisito no funcional corresponde a la etapa de *análisis*, ya que en esta etapa es donde se definen ciertos criterios como el mínimo, promedio y máximo número de usuarios concurrentes que puede soportar el sistema, teniendo en cuenta cuántas solicitudes se pueden procesar simultáneamente en el sistema, si la falta de capacidad no es detectada a tiempo esto podría traer consecuencias al sistema, por esta razón debemos saber cuánta información y por cuánto tiempo puede ser almacenada por el sistema, etc.

Ejemplo y evidencia de aplicación

Primero es prudente ejemplificar este requisito no funcional con una situación externa al área de los sistemas, imaginando una pizzería en la cual las órdenes se escriben en una nota y se pegan horizontalmente, en orden de llegada, a la pared de la cocina en la ventana donde se reciben las órdenes. Si se reciben demasiadas órdenes muy rápido, en algún momento se va a llegar a un límite en el que las notitas no cabrán más en la pared y no se podrán recibir más órdenes por limitaciones fuera de las capacidades de los cocineros.

Ahora bien, en el área de sistemas, se puede ver fácilmente traducido el ejemplo anterior, donde una aplicación de pizzas se encarga de recibir los pedidos de pizzas y proporcionarle al usuario un seguimiento en vivo del estado de su pizza. Si algún día se reciben más pedidos de los que en la etapa de análisis se habían planeado recibir, los servidores no soportan recibir más peticiones o simplemente fallará al intentar proporcionar el seguimiento.

Qué es un patrón arquitectónico?
Busca establecer un esquema de funcionamiento y organización de un sistema, criterios y restricciones para todos los componentes, responsabilidades dentro de este y sus interrelaciones.

Cómo implementarlo dentro de un patrón arquitectónico
Este requisito se hace visible en el patrón arquitectónico orientado a servicios, donde el software que hace de servidor, controla la concurrencia de las solicitudes que hacen los usuarios, es decir, el procesamiento simultáneo de diferentes solicitudes. El requisito capacidad se implementa en el control de las sesiones y cómo se procesarán estas solicitudes en hilos diferentes o en otros métodos de procesamiento concurrente.

Conclusiones

Con esta investigación, se dio a conocer la importancia de los requisitos no funcionales a la hora de realizar un proyecto siendo estos un soporte para que el proyecto tenga buenos estándares de calidad e indicadores que ayudan a mejorar sustancialmente la eficiencia del producto final. Además, podemos observar que a través de la revisión y especificación de los requisitos no funcionales se logran captar importantes detalles específicos de lo que debería hacer el sistema en desarrollo, reconociendo que los requisitos no funcionales llegan a ser un factor muy importante a tener en cuenta a lo largo de las diferentes etapas de desarrollo, permitiendo ampliar la calidad que puede tener un software al implementar pequeñas prácticas.

Referencias

[1] |https://www.nngroup.com/articles/website-response-times/
[2]https://www.altexsoft.com/blog/non-functional-requirements/
[3] https://octoperf.com/blog/2019/06/26/non-functional-requirements/
[4] https://www.ecured.cu/Requisitos_no_funcionales
[5]http://acceleratteddevelopment.blogspot.com/2013/12/what-heck-are-non-functional.html
[6]https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a
[7] https://www.ecured.cu/Requisitos_no_funcionales
[8]https://www.sic.gov.co/sites/default/files/normatividad/Proyecto_Resolucion_Reglamenta_Actividad_del_Avaluador_Ley_1673_Anexo5.pdf
[9] https://octoperf.com/blog/2019/06/26/non-functional-requirements/

[10]https://www.mineducacion.gov.co/1759/articles-310035_archivo_pdf_cm382012_anexo3.pdf
[11] https://www.altexsoft.com/blog/non-functional-requirements/
[12] https://www.oreilly.com/library/view/mastering-non-functional-requirements/9781788299237/cea87428-8d1d-46f7-9caf-6951ea3d9645.xhtml
[13] https://seilevel.com/requirements/non-functional-requirements-interoperability-requirements#:~:text=In_the_sea_of_non,other_systems_and_external_hardware.&text=The information was not being,receiving system from processing it.
[14] https://www.oreilly.com/library/view/mastering-non-functional-requirements/9781788299237/6c43449b-b923-40b5-810f-1710544d9de1.xhtml
[15] https://qa-platforms.com/understanding-non-functional-requirements/
[16] http://www.julianbrowne.com/article/nfrs