Informe de la investigación realizada sobre algunos requisitos no funcionales

Profesor
Róbinson Coronado García

Alumnos
Oscar Darío Botero Vargas
C.C. 71’764,308

Kelly Stefanía Tobón Montoya
C.C. 1,038,814,926

Juan Guillrmo Hernádez Alarcón
C.C. 1,028,034,458

Universidad de Antioquia
Curso de Arquitectura del Software
Medellín
Septiembre de 2020

Según el artículo *Requerimientos Funcionales y No Funcionales, ejemplos y tips* (20 de abril de 2018), los requisitos NO funcionales son los que:

> ...no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad. En palabras más sencillas, no hablan de "lo que" hace el sistema, sino de "cómo" lo hace.

Al respecto, el artículo *Non-functional requirement* (11 de septiembre de 2020) también dice que:

> En la ingeniería de sistemas y la ingeniería de requisitos, un requisito no funcional (RNF) es un requisito que especifica criterios que pueden usarse para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos. Se contrastan con los requisitos funcionales que definen comportamientos o funciones específicas. El plan para implementar los requisitos funcionales se detalla en el *diseño del sistema*. El plan para implementar los requisitos no funcionales se detalla en la *arquitectura del sistema*, porque normalmente son requisitos arquitectónicamente importantes.

A continuación presentamos 7 requisitos no funcionales.

Adaptabilidad

El artículo *Adaptability* (29 de abril de 2020) dice que:

> La adaptabilidad debe entenderse aquí como la capacidad de un sistema (por ejemplo, un sistema informático) para adaptarse de manera eficiente y rápida a las circunstancias cambiantes. Por lo tanto, un sistema adaptativo es un sistema abierto que puede adaptarse a su comportamiento de acuerdo con los cambios en su entorno o en partes del propio sistema. Es por ello que se puede expresar el requisito de reconocer la demanda de cambio sin ningún otro factor involucrado.

Transparencia

El artículo *Transparency (behavior)* (8 de julio de 2020) dice que:

> En el mundo del software informático, el software de código abierto se refiere a la creación de software, al que se puede acceder libremente al código fuente subyacente. Esto permite su uso, estudio y modificación sin restricciones.
>
> En seguridad informática, el debate continúa en cuanto a los méritos relativos de la divulgación completa de las vulnerabilidades de seguridad, frente a un enfoque de seguridad por oscuridad.
>
> Hay un sentido diferente (quizás casi opuesto) de transparencia en la interacción humano-computadora, por el cual un sistema después del cambio se adhiere a su interfaz externa anterior tanto como sea posible mientras cambia su comportamiento interno. Es decir, un cambio en un sistema es transparente para sus usuarios si el cambio es imperceptible para ellos.

Legibilidad

Según el artículo *Readability of source code* (14 de septiembre de 2020):

En la programación de computadoras, la legibilidad se refiere a la facilidad con la que un lector humano puede comprender el propósito, el flujo de control y el funcionamiento del código fuente. Afecta [los aspectos de calidad de] la portabilidad, la facilidad de uso (usability) y, lo más importante, la capacidad de mantenimiento.

La legibilidad es importante porque los programadores pasan la mayor parte de su tiempo leyendo, tratando de comprender y modificar el código fuente existente, en lugar de escribir código fuente nuevo. El código ilegible a menudo conduce a errores, ineficiencias y código duplicado. Un estudio encontró que unas pocas transformaciones simples de legibilidad acortaron el código y redujeron drásticamente el tiempo para comprenderlo.

Seguir un estilo de programación coherente a menudo ayuda a mejorar la legibilidad. Sin embargo, la legibilidad es más que sólo estilo de programación. Muchos factores, que tienen poco o nada que ver con la capacidad de la computadora para compilar y ejecutar el código de manera eficiente, contribuyen a la legibilidad. Algunos de estos factores incluyen:

- Diferentes estilos de sangría (espacios en blanco)
- Comentarios
- Descomposición
- Convenciones de nomenclatura para objetos (como variables, clases, procedimientos, etc.)

Los aspectos de presentación de esto (como sangrías, saltos de línea, resaltado de color, etc.) a menudo son manejados por el editor de código fuente, pero los aspectos de contenido reflejan el talento y las habilidades del programador.

También se han desarrollado varios lenguajes de programación visual con la intención de resolver los problemas de legibilidad mediante la adopción de enfoques no tradicionales para la estructura y visualización del código. Los entornos de desarrollo integrados (IDE) tienen como objetivo integrar toda esa ayuda. Técnicas como la refactorización de código pueden mejorar la legibilidad.

Entorno de Desarrollo

Según el artículo *Deployment environment* (5 de mayo de 2020):

El entorno de desarrollo (dev) es el ambiente en el cual los cambios al software son desarrollados, la más de las veces es simplemente la estación de trabajo de un desarrollador individual. Esto difiere, de varias maneras, del entorno de destino final pues puede no ser una computadora de escritorio (puede ser un teléfono inteligente, un sistema integrado, una máquina bruta en un centro de datos, etc.) y si sí lo es, el entorno del desarrollador incluirá herramientas de desarrollo como un compilador, un entorno de desarrollo integrado (IDE), versiones diferentes o adicionales de bibliotecas y software de apoyo, etc., que no están presentes en el entorno de un usuario.

En el contexto del control de revisiones, particularmente con varios desarrolladores, se establecen distinciones más precisas: un desarrollador tiene una copia de trabajo del código fuente en su máquina y envia los cambios al repositorio, cuyo destino es el tronco o una rama, según la metodología de desarrollo. El entorno de una estación de trabajo individual, en el que se trabajan y prueban los cambios, puede denominarse *entorno local*, *entorno de pruebas* o *caja de arena* (sandbox). La creación de la copia del repositorio con el código fuente en un entorno

limpio es un paso separado, parte de la integración (integración de cambios dispares), y este entorno puede denominarse *entorno de integración* o *entorno de desarrollo*; en la integración continua esto se hace con frecuencia, tan a menudo como por cada revisión. El concepto a nivel de código fuente de "confirmar un cambio en el repositorio", seguido de la construcción del tronco o rama, corresponde a presionar para liberar de local (entorno del desarrollador individual) a integración (construcción limpia); un lanzamiento incorrecto en este paso significa que un cambio rompió la compilación, y revertir el lanzamiento corresponde a revertir todos los cambios desde ese punto en adelante, o deshacer sólo el cambio de ruptura, si es posible.

Soportabilidad

Según el artículo *Conceptos de Confiabilidad, Mantenibilidad y Soportabilidad* (19 de abril de 2017):

> Se denomina soportabilidad a la probabilidad de poder atender una determinada solicitud de mantenimiento en el tiempo de espera prefijado y bajo las condiciones planeadas. También se puede referir a la soportabilidad como la velocidad de respuesta de mantenimiento y su influencia sobre el mantenimiento proactivo.

Según el artículo *Capturing Architectural Requirement*s. (15 de noviembre de 2005):

> La capacidad de soporte se refiere a características tales como capacidad de prueba, adaptabilidad, mantenibilidad, compatibilidad, configurabilidad, instalabilidad, escalabilidad y localización.

Compatibilidad

Según el artículo *Requisitos no funcionales*:

> La compatibilidad es la capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:
>
> - Coexistencia: Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
>
> - Interoperabilidad: Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.
>
> Ejemplo: El sistema deberá ser compatible con los navegadores Internet Explorer 6 y Mozilla Firefox 3.

Exactitud o precisión

Según el artículo *What are Accuracy Requirements?* (02 de mayo de 2018):

> No existe una definición estándar de un requisito no funcional de precisión. Se definirá para cada proyecto donde sea necesario especificarlo. Este principio es válido para todos los requisitos no funcionales.
>
> Un punto general sobre los requisitos de precisión que, en principio, se aplica a todos los requisitos de todos modos: a menudo, el requisito de precisión se indicará como "La solución debe ser 100% precisa".

Ser 100% exacto puede implicar una gran cantidad de verificación y doble verificación (para garantizar que los datos sean correctos), lo que puede hacer que entren en conflicto con otros requisitos relacionados con el tiempo permitido que debe tomar un proceso determinado. Estos requisitos también deben descubrirse y, como ocurre con cualquier requisito, comprobar que no están en conflicto con otros requisitos, en este caso como la precisión. Depende del Analista de Negocios del proyecto mediar en la resolución de los requisitos en conflicto (por ejemplo, en este caso, lograr el acuerdo de que la precisión se esforzará por ser del 100%, pero que esto no se puede garantizar en el caso de (digamos) registrar los datos de un cliente de nacimiento: la verificación necesaria para garantizar que sea 100% precisa el 100% del tiempo para todos los clientes sería complicada, costosa y llevaría mucho tiempo).

Registrabilidad

Según el artículo *Architectural Requirements*:

> Se trata de un requerimiento no funcional que está en el grupo de operabilidad (operability). Nos indica básicamente la capacidad de registrar logs como forma de seguimiento a las transacciones, operaciones o procesos que realiza un sistema.
>
> La registrabilidad puede ser tan sencilla como se nos ocurra o tan compleja como la necesitemos por tanto no es requerimiento a pasar por alto aunque no es obligatorio. En la actualidad existen librerías y frameworks que nos ayudan con el registro de logs de nuestras aplicaciones o sistemas.

Understandability

Según el documento *Non-Functional Requirements* (12 de marzo de 2008):

> La capacidad del producto de software para permitir que el usuario comprenda si el software es adecuado y cómo se puede utilizar para tareas y condiciones de uso particulares. Este RNF puede verse relacionado con lo intuitivo y lo adaptable, es decir, el software se auto explica y es evidente lo que reduce el esfuerzo de aprendizaje para quién lo use, además es flexible al usuario y a las necesidades cambiantes.

Referencias

Requerimientos Funcionales y No Funcionales, ejemplos y tips. (20 de abril de 2018). En *medium.com*. Recuperado el 16 de septiembre de 2020 desde https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Non-functional requirement. (11 de septiembre de 2020). En *Wikipedia*. Recuperado el 16 de septiembre de 2020 desde https://en.wikipedia.org/w/index.php?title=Non-functional_requirement&oldid=977899249

Adaptability. (29 de abril de 2020). En *Wikipedia*. Recuperado el 16 de septiembre de 2020 desde https://en.wikipedia.org/w/index.php?title=Adaptability&oldid=953823158

Transparency (behavior). (8 de julio de 2020). En *Wikipedia*. Recuperado el 16 de septiembre de 2020 desde https://en.wikipedia.org/w/index.php?title=Transparency_(behavior)&oldid=966685840

Readability of source code. (14 de septiembre de 2020). En *Wikipedia*. Recuperado el 16 de septiembre de 2020 desde https://en.wikipedia.org/w/index.php?title=Computer_programming&oldid=978364414#Readability_of_source_code

Deployment environment. (5 de mayo de 2020). En *Wikipedia*. Recuperado el 16 de septiembre de 2020 desde https://en.wikipedia.org/w/index.php?title=Deployment_environment&oldid=955042268#Development

Conceptos de Confiabilidad, Mantenibilidad y Soportabilidad. (19 de abril de 2017). En *Scribd*. Recuperado el 18 de septiembre de 2020 desde https://es.scribd.com/document/345774524/Conceptos-de-Confiabilidad-Mantenibilidad-y-Soportabilidad

Capturing Architectural Requirements. (15 de noviembre de 2005) En *IBM Corp*. Recuperado el 18 de septiembre de 2020 desde https://www.ibm.com/developerworks/rational/library/4706.html

Requisitos no funcionales. En *Libro digital Modelado UML en Ingeniería del Software.* Recuperado el 18 de septiembre de 2020 desde https://personales.unican.es/blancobc/apuntesis/modeladoUML/RNF/index.html

Especificación de Requisitos del Sistema. (1 de marzo de 2013) En *Marco de Desarrollo de la Junta de Andalucía (MADEJA)*. Recuperado el 18 de septiembre de 2020 desde http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407

Non-Functional Requirements. (12 de marzo de 2008) En *Marco de Desarrollo de la Junta de Andalucía (MADEJA)*. Recuperado el 18 de septiembre de 2020 desde https://www.site.uottawa.ca/~bochmann/SEG3101/Notes/SEG3101-ch3-4%20-%20Non-Functional%20Requirements%20-%20Qualities.pdf

Architectural Requirements. (30 de abril de 2004) IBM. Recuperado el 18 de septiembre de 2020 desde https://www.ibm.com/developerworks/rational/library/4708.html

What are Accuracy Requirements? (02 de mayo de 2018) Requirements. Recuperado el 18 de septiembre de 2020 desde https://requirements.com/Content/What-is/what-are-accuracy-requirements#:~:text=For%20the%20purposes%20of%20this,will%20record%20or%20produce%20data.