

11-3-2020

Determinación modelo de calidad

Mentral

Alumno: Norton Irarrázabal

Correo: [norton.dante.i@gmail.com](mailto:norton.dante.i@gmail.com)

Docente: Guillermo Leyton

Asignatura: Software II

Contenido

[Propósito 1](#_Toc35189018)

[Determinación 1](#_Toc35189019)

[Enfoque ontológico 5](#_Toc35189020)

[Implicancia 6](#_Toc35189021)

# Propósito

Este documento tiene como finalidad determinar los atributos de calidad del software Mentral, primero realizando un filtro mediante los modelos de calidad como McCall, Dromey, Furps, ISO 9126 vistos en clases y posteriormente realizar un modelo de calidad desde el enfoque ontológico.

Por otra parte, esto es útil para determinar los objetivos durante el proceso de desarrollo ya que estos se establecen en base a la calidad del producto de software.

# Determinación

La relevancia (r) de un atributo se calificará del 0 (menor relevancia) al 4 (mayor relevancia).

Tipos de atributos de calidad:

* Externos: Aquellos definidos a partir de la visión del usuario del software.
* Internos: Visión de calidad técnica, determinado por el equipo de desarrollo, no visible para el usuario común.

Consideraciones:

* Dominio del software:
  + Software educativo.
  + Es un software de tutoría inteligente.
  + Los usuarios serán estudiantes universitarios, 18+ años de edad.
  + Principalmente abarca carreras del área de la salud.
  + Los estudiantes buscan softwares fáciles de usar, eficientes, intuitivos y que despierten sus intereses saliendo de lo común.
  + Los estudiantes buscan que software tengan buen aspecto visual, que sean llamativos y estructurado.
  + El software debe ser capaz de ir guardando el progreso de los sujetos que lo utilicen, ya que el no hacerlo generaría rechazo.
  + Los usuarios buscan que el software este siempre disponible para cuando sientan motivación de estudio.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Descripción | Tipo | r | Argumento |
| Disponibilidad | Hace referencia al porcentaje de tiempo durante el cual se puede acceder y utilizar el sistema. | Externo | 4 | Este atributo es muy importante ya que los usuarios tienen motivación de estudio de forma aleatoria, por lo que el software debe estar disponible cuando ello ocurra, en cualquier espacio de tiempo. |
| Confidencialidad | Grado en el que los datos tienen atributos que aseguran que los datos son sólo accedidos e interpretados por usuarios autorizados en un contexto de uso específico. | Externo | 4 | El usuario debe tener plena confianza en que sus datos personales NO son inaccesibles por personas no autorizadas, resguardando su privacidad. |
| Funcionalidad | Habilidad del sistema para realizar el trabajo para el cual fue concebido | Externo | 4 | El sistema debe cumplir con el modelo y con la meta propuesta, ya que esto es la razón de su existencia.  El no cumplimiento de sus funciones supone el fracaso del software. |
| Eficiencia | Es un indicador de la capacidad de respuesta del sistema responde a un evento o estímulo en un intervalo de tiempo dado. | Interno/Externo | 4 | Los usuarios buscan software que sean rápidos de usar con pocas transiciones, pantallas de carga, respuestas lentas, etc.  Si el software responde lento el usuario pierde interés. Por otra parte, del punto de vista del desarrollador una eficiencia lenta también reduce su velocidad de desarrollo y realización de pruebas. |
| Fiabilidad | Se refiere a la capacidad del sistema de funcionar permanentemente sin fallos. | Externo | 4 | Resulta relevante para el usuario que el sistema funcione sin fallos, para no generar interrupciones, además a nivel lógico debe ser preciso.  Cabe resaltar que en caso de un fallo debe permitir continuar con el funcionamiento y conservar la integridad de los datos. |
| Interoperabilidad | Que tan bien puede el sistema intercambiar datos con otros sistemas. | Interno | 1 | El software únicamente intercambiara datos con la BD. |
| Integridad | Define la consistencia y coherencia del diseño total.  A nivel conceptual: Esto incluye la forma en que los componentes o módulos han sido diseñados, así como factores como el estilo de codificación y la nomenclatura de las variables.  A nivel de datos: Los datos se deben mantener íntegros y sin alteraciones. | Interno | 4 | Atributo de alta relevancia por parte del desarrollador ya que trabajar bajo patrones de arquitectura, codificación y nomenclaturas definidas implica agilizar el proceso de programación y mitigar los errores, además permite que la lógica del software sea mas comprensible a nivel lógico.  A nivel de datos se debe considerar que estos no puedan perderse o modificarse de forma no intencionada ya que afectaría al progreso del usuario. |
| Mantenibilidad | Habilidad de un sistema para permitir cambios en sus componentes, servicios, características e interfaces en la medida en que dichos cambios son requeridos cuando se adiciona o cambia la funcionalidad, se corrigen errores o se suplen nuevos requerimientos de negocios. | Interno | 4 | Muy importante ya que este es el medio por el cual se realizan pruebas y cambios permitiendo mayor flexibilidad. |
| Portabilidad | Habilidad de un sistema para ejecutar en diferentes ambientes (hardware, software, o una combinación de ambos). | Interno/Externo | 0 | El producto es un software es de escritorio que se desenvolverá concretamente en el S.O Windows.  Por otra parte, debe ser capaz de ejecutarse en hardware de bajo rendimiento. |
| Reusabilidad | Define la capacidad de un componente y un subsistema para ser usado por otras aplicaciones en otros escenarios. | Interno | 0 | Los componentes del software no se reutilizarán para otras aplicaciones o sistemas diferentes. |
| Escalabilidad | Es la capacidad de la aplicación para funcionar correctamente si el tamaño de procesamiento se incrementa. | Interno/externo | 2 | El software debe ser capaz de mantener su rendimiento en caso de incremento de procesamiento, ya que la fluidez es un aspecto relevante para el usuario, así como también para el desarrollador. |
| Capacidad de prueba | Que tan fácil es realizar una prueba en el sistema. | Interno | 4 | Importante para el desarrollador ya que permite corroborar de forma rápida el aspecto lógico del software, mitigando y corrigiendo errores. |
| Facilidad de uso | Que tan fácil un usuario puede hacer las tareas en el sistema. | Externo | 4 | De gran importancia ya que el usuario busca facilidad de aprender a usar una nueva herramienta, aprendiendo de forma intuitiva como usarlo. También este debe resultar atractivo para los usuarios. |
| Flexibilidad | Es la habilidad del sistema para adaptarse a ambientes, situaciones variables, para soportar cambios en políticas de negocios y reglas de negocio. | Interno | 3 | Durante el desarrollo del software se realizan variados cambios producto de que la problemática en si puede abarcarse de distintas maneras, por lo que resulta beneficioso poder adaptar el sistema a nuevos requerimientos. |

Los atributos considerados relevantes para el software son los siguientes:

* Disponibilidad.
* Confidencialidad.
* Funcionalidad.
* Eficiencia.
* Fiabilidad.
* Integridad.
* Mantenibilidad.
* Capacidad de prueba.
* Facilidad de uso.
* Flexibilidad.

# Enfoque ontológico

Kesh divide en dos secciones la calidad, por un lado tenemos el funcionamiento y por otro la ontología en donde sus componentes ontológicos se describen de la siguiente forma:

|  |  |
| --- | --- |
| Componentes | Descripción |
| Adecuación | Representa la capacidad para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. |
| Solidez | Se refiere a la capacidad del software de defenderse de las acciones anormales (analizar, condicionar, limitar) que llevan al sistema a un estado no previsto, causando un comportamiento inesperado, indeseado y posiblemente erróneo. |
| Consistencia | Un software es consistente cuando los requerimientos no poseen definiciones contradictorias. Las especificaciones deben ser coherentes a través del diseño y las técnicas de documentación durante el desarrollo. |
| Concisión | Compacto que resulta ser el programa en términos de líneas de código. Capacidad de ser conciso en el diseño. |
| Compleción | Significa que todos, los servicios solicitados por el usuario están definidos. El modelo contiene todas las sentencias acerca del dominio que son correctas y relevantes. |
| Cohesión | Se refiere al grado en el que los elementos de un modulo permanecen juntos, la relación entre las piezas de funcionalidad dentro de un módulo dado. (A mayor cohesión menor acoplamiento[[1]](#footnote-1) - Una clase con alta cohesión suele cumplir el principio de única responsabilidad[[2]](#footnote-2).) |
| Validez | Es comprobar que el sistema software descrito por la ERS se corresponde con las necesidades de negocio de clientes y usuarios. Para ello no basta con preguntar a un único usuario, ya que todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos. |

Una ves descritos podemos percatarnos que la ontología a diferencia de las “listas” de calidad va más allá de los atributos inherentes al software, sino que también implica el proceso de desarrollo y los factores internos como externos.

# Implicancia

Kesh además especifica que la ontología afecta a la funcionalidad, sin embargo, la implicancia no puede ser medida de forma binaria por ende utilizamos lógica difusa para representar esta situación:

|  |  |  |  |
| --- | --- | --- | --- |
| Concepto | Variable lingüística | Universo normalizado | Función de membresía |
| Implicancia | Alta | 0.75<x<1 | Triangular |
| Media | 0.25<x<0.75 | Trapezoidal |
| Baja | 0<x<0.25 | Triangular |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Calidad | | | | | | | | |
|  | | | | Funcionamiento | | | | |
| S\_1 | S\_2 | S\_3 | S\_4 | S\_5 |
| Usabilidad (usuario) | Usabilidad (diseñador) | Mantenibilidad | Precisión | Rendimiento |
| Ontología | Estructura | O\_1 | Adecuación | 1 | 0 | 0 | 0 | 0 |
| O\_2 | Solidez | 0 | 1 | 1 | 0 | 0 |
| O\_3 | Consistencia | 0.7 | 0.9 | 0 | 0.9 | 0 |
| O\_4 | Concisión | 0.5 | 0 | 1 | 0 | 0.8 |
| Contenido | O\_5 | Compleción | 0.7 | 0.9 | 0 | 0.9 | 0.6 |
| O\_6 | Cohesión | 0 | 0.9 | 0.8 | 0 | 0 |
| O\_7 | Validez | 0 | 0.5 |  | 0 | 0 |

Posteriormente se determina los valores de atributos funcionales en base a los componentes ontológicos.

S\_1 = O\_1\*f(1)+O\_3\*f(0.7)+O\_4\*f(0.5)+O\_5\*f(0.7)

S\_2 = O\_2\*f(1)+O\_3\*f(0.9)+O\_5\*f(0.9)+O\_6\*f(0.9)+O\_7\*f(0.5)

S\_3 = O\_2\*f(1)+O\_4\*f(1)+O\_6\*f(0.8)

S\_4 = O\_3\*f(0.9)+O\_5\*f(0.9)

S\_5 = O\_4\*f(0.8)+O\_5\*f(0.6)

Los valores de los componentes ontológicos se obtienen mediante métricas que evalúen la calidad de un componente respecto de la funcionalidad a la que este aplicando.

1. **Acoplamiento**: Nivel de interdependencia entre módulos de software, una medida de qué tan cercanamente conectados están dos rutinas o módulos de software, Un ejemplo simple de acoplamiento es cuando un componente accede directamente a un dato que pertenece a otro componente. En ese caso, el resultado del comportamiento del componente A dependerá del valor del componente B, por lo tanto, están acoplados. [↑](#footnote-ref-1)
2. **Principio de responsabilidad única**: Establece que cada módulo o clase debe tener responsabilidad sobre una sola parte de la funcionalidad proporcionada por el software y esa responsabilidad debe estar completamente encapsulada por la clase, módulo o función. [↑](#footnote-ref-2)