Modelo de Calidad

SENA

FarmiFarmacy

Andrés Felipe Monroy Moreno

Brayan Andrés Puello Sanches

Heelen Lizeth Cano Moreno

Bogotá D.C

Fecha de elaboración:

Contenido

[1. Planteamiento del problema 3](#_Toc95855042)

[1.2 Problema general 4](#_Toc95855043)

[2. Objetivo 4](#_Toc95855044)

[2.1 Objetivo general 4](#_Toc95855045)

[2.2 Objetivo especifico 5](#_Toc95855046)

[3. Alcance y/o requisitos de evaluación 5](#_Toc95855047)

[3.1 Tener un modelo de calidad totalmente funcional 5](#_Toc95855048)

[3.2 Garantizar el ciclo de vida del software 5](#_Toc95855049)

[Anexo 1. Cronograma de requerimientos 6](#_Toc95855050)

[Anexo 2. Cronograma de pruebas 6](#_Toc95855051)

[4. Elementos para evaluar 6](#_Toc95855052)

[4.1 Métricas Externas e Internas 6](#_Toc95855053)

[4.1.1 Características 6](#_Toc95855054)

[4.2 Sub Característica 7](#_Toc95855055)

[4.3 Calidad de uso 8](#_Toc95855056)

[4.4 Características razón de implementación 8](#_Toc95855057)

[4.5 Calidad de uso porque implementar 9](#_Toc95855058)

[5. Métodos de evaluación 10](#_Toc95855059)

[5.1.1 Evaluación de características y subcaracterísticas 10](#_Toc95855060)

[6. Diseño de la evaluación 11](#_Toc95855061)

[Referencias 13](#_Toc95855062)

# Planteamiento del problema

Se ve muy a menudo que las diferentes industrias de software cuando se realiza un proyecto de software y se empiezan a hacer actualizaciones o pequeños cambios comienzan los errores en software donde ya publicado y con un reconocimiento, puede que sea peligroso tanto para el cliente y los propios dueños donde se pueden empezar a filtrar información y/a perder la información, donde genera inseguridad un ciclo de vida de muy corto para el software sé que en Colombia las principales fallas según Software ONE y otras fuentes en 2020 en Colombia se vi que los errores más comunes a la hora de desarrollar un software son (fallas de almacenamiento, perdida de bases de datos, violaciones de seguridad, fallas en la programación, desastres naturales, falta de actualizaciones, falta de conocimiento, y falta de un modelo de calidad).

En la historia podemos ver como una cosa tan pequeña conto la vida de muchas personas y no solo eso si no gastos económicos muy altos como en Mariner 1.

En el año 1962, la Nasa se disponía a realizar el lanzamiento al espacio de la misión Mariner 1, con el fin de navegar la órbita de Venus. Un error de programación representó la diferencia entre el éxito y una catástrofe total.

La Mariner 1 fue creada para recabar datos sobre la temperatura y atmósfera de Venus, pero no logró salir de la atmósfera de la tierra. La catástrofe se originó a solo 5 minutos del despegue debido a la mala transcripción de un código y la omisión de un guion “-“.

El fracaso de la Mariner 1 produjo la pérdida de 18,5 millones de dólares a la Nasa.

Por esta razón, según el estándar ISO/IEC 25000 (2005), la calidad del producto no solamente

debe ser vista como una actividad exclusiva de la fase de pruebas de software,

sino también debe considerar la evaluación de los entregables del producto

durante su ciclo de vida.

La problemática se resume en:

* Errores humanos.
* Baja calidad de software.
* Falta de evaluar la capacidad y la madurez de los procesos.
* Sobrecostos debido a asignación de recursos para solucionar errores de software y retrasos en los proyectos de software.

## 1.2 Problema general

¿Cómo nos puede ayudar un modelo de calidad a mejorar el software?

# Objetivo

## 2.1 Objetivo general

Basado en el modelo de calidad ISO/IEC 25000 es mejorar todos los procesos del software realizando sus evaluaciones

## 2.2 Objetivo especifico

Los módulos los cuales van a hacer evaluados son todos aquellos que componen y nos ayudador a desarrollar el mismo, se entregara un informe y se evidenciara el método de evaluación que se tomo con cada una de las partes donde se tendrá en cuenta, los requisitos funcionales del software también se realizar pruebas al software de igual manera se entregara totalmente documenta así mismo se evitaran posibles errores.

# Alcance y/o requisitos de evaluación

## 3.1 Tener un modelo de calidad totalmente funcional

El modelo de calidad que se realizara tendré que cumplir con todo lo anterior propuesto teniendo presente el cronograma de requerimientos funcionales del software adicional de eso tendremos presentes los informes de pruebas para mayor seguridad y certeza de lo que se analizó en el software y que en el momento de ser entregado cumpla con las expectativas y lo esperado del cliente.

## 3.2 Garantizar el ciclo de vida del software

Se busca tener una vida optima del software donde cada parte funcione según o mejor lo planteado que este puede ser actualizado sin tener ningún error y no solo eso si no que sea totalmente funcional a los requerimientos del cliente.

3.3 Edificar las partes que serán evaluadas

Estas serán cada uno de los requerimientos funcionales que se tienen en el software y que cumpla en un 80% o 90% las expectativas de las evaluaciones también se tendrá presente las pruebas de caja negra, caja blanca y las pruebas unitarias para garantizar la seguridad de este.

En resumen, lo que se busca en el alcance es:

* Mejoras en la usabilidad
* Satisfacción de los usuarios
* Aporte en la gestión del tiempo
* Software libre de errores

### Anexo 1. Cronograma de requerimientos

### Anexo 2. Cronograma de pruebas

# Elementos para evaluar

## 4.1 Métricas Externas e Internas

### 4.1.1 Características

|  |  |
| --- | --- |
| **Característica** | **Definición** |
| **Funcionalidad** | La capacidad del producto software para proveer las funciones que satisfacen las necesidades explícitas e implícitas cuando el software se utiliza bajo condiciones específicas. |
| **Fiabilidad** | La capacidad del producto software para mantener un nivel especificado de funcionamiento cuando se está utilizando bajo condiciones especificadas. |
| **Eficiencia** | La capacidad del producto software para proveer un desempeño apropiado, de acuerdo con la cantidad de recursos utilizados y bajo las condiciones planteadas. |
| **Facilidad de mantenimiento** | Capacidad del producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y en requerimientos y especificaciones funcionales. |

Nota elaboración: ( <https://sites.google.com/site/informaticamcprats/iso-9126>)

## 4.2 Sub Característica

|  |  |  |
| --- | --- | --- |
|  | **Sub Característica** | **Descripción** |
| **Funcionalidad** | Aplicabilidad | La capacidad del producto software para proveer un conjunto apropiado de funciones para las tareas y objetivos especificados por el usuario. |
| Precisión | La capacidad del producto software para proveer los resultados o efectos acordados con un grado necesario de precisión. |
| Seguridad | La capacidad del producto software para proteger la información y los datos de modo que las personas o los sistemas no autorizados no puedan leerlos o modificarlos y a las personas o sistemas autorizados no se les denegará el acceso. |
| Conformidad           de       la  funcionalidad | La capacidad del producto software de adherirse a los estándares, convenciones o regulaciones legales y prescripciones similares referente a la funcionalidad. |
| **Fiabilidad** | Madurez | La capacidad del producto software para evitar fallas como resultado de errores en el software. |
| Tolerancia a fallos | La capacidad del producto software para mantener un nivel especificado de funcionamiento en caso de errores del software o de incumplimiento de su interfaz especificada. |
| Recuperabilidad | La capacidad del producto software para reestablecer un nivel especificado de funcionamiento y recuperar los datos afectados directamente en el caso de una falla. |
| **Eficiencia** | Comportamiento           en       el  tiempo | La capacidad del producto software para proveer tiempos apropiados de respuesta y procesamiento, y ratios de rendimiento cuando realiza su función bajo las condiciones establecidas. |
| **Facilidad de Mantenimiento** | Analizabilidad | La capacidad del producto software para ser diagnosticado por deficiencias o causas de fallas en el software o la identificación de las partes a ser modificadas. |
| Confiabilidad | La capacidad del software para permitir que una determinada modificación sea implementada. |
| Estabilidad | La capacidad del producto software para evitar efectos inesperados debido a modificaciones del software. |
| Testeabilidad | La capacidad del software para permitir que las modificaciones puedan ser validadas. |
| Conformidad de facilidad de mantenimiento | La capacidad del software para adherirse a estándares o convenciones relativas a la facilidad de mantenimiento. |

Nota elaboración: (<https://sites.google.com/site/informaticamcprats/iso-9126>)

## 4.3 Calidad de uso

|  |  |
| --- | --- |
| **Característica** | **Definición** |
| **Efectividad** | La capacidad del producto software para permitir a los usuarios lograr las metas especificadas con precisión y completitud en un contexto de uso específico. |
| **Productividad** | La capacidad del producto software para permitir a los usuarios emplear cantidades apropiadas de recursos en relación con la efectividad lograda en un contexto de uso específico. |
| **Satisfacción** | La capacidad del producto software para satisfacer a los usuarios en un contexto de uso específico. |

Nota elaboración: (<https://sites.google.com/site/informaticamcprats/iso-9126>)

## 4.4 Características razón de implementación

|  |  |  |
| --- | --- | --- |
|  | Razón de uso | Subcaracteriticas |
| **Funcionalidad** | Este es muy importante para el funcionamiento eficaz del software porque si no tenemos un software funcional con una buena seguridad no podemos tener una programación completa. | * Aplicabilidad * Precisión * Seguridad * Conformidad de la   funcionalidad |
| **Fiabilidad** | Si no tenemos un software fiable capaz de recuperarse de fallo es complicada detener un ciclo de vida optimo y una durabilidad en el mercado es muy importante tener siempre un respaldo una prueba para cada error que se pueda presentar. | * Madurez * Tolerancia a fallos * Recuperabilidad |
| **Eficiencia** | La idea del grupo a desarrollar es tener siempre presente que los tiempos estimados sean cumplidos para que el cliente tenga una confiabilidad en el software que se implementa con contratiempos será muy difícil para el cliente tener una óptima experiencia y no solo eso si no que le tocaría esperar mucho más de lo que se tenía planeado. | * Comportamiento en el   tiempo |
| **Facilidad de mantenimiento** | Vemos necesario tener un manteniendo completo donde genere confiabilidad a los usuarios del software es importante para nosotros resaltar estos puntos para tener un óptimo funcionamiento. | * Analizabilidad * Confiabilidad * Estabilidad |

Nota elaboración: (Elaboración propia)

## 4.5 Calidad de uso porque implementar

|  |  |  |
| --- | --- | --- |
| **Característica** | Razón | Métricas |
| **Efectividad** | Con esta característica veremos el cumplimiento de un gran parte del software ya que veremos aquello requerimientos de usuario siendo aprovechados al máximo en el desarrollo. | Efectividad laboral. |
| **Productividad** | Siempre se tiene en cuenta que entre más rápido se entregue el trabajo mejor para todo nos ahorramos tiempo y lo mejor de todo tenemos más tiempo para poder hacer pruebas extras en caso de errores. | Horas extras. |
| **Satisfacción** | Esta parte complementa la efectividad porque si cliente le gusta lo que y lo que tiene se abra logra cumplir las expectativas de lo que el pago. | Buena experiencia |

Nota elaboración: (Elaboración propia)

# Métodos de evaluación

5.1 Niveles de importancia

Este se será el que evaluará la importancia de cada una de las características y subcaracterísticas y se dará el motivo de su evaluación

|  |  |  |  |
| --- | --- | --- | --- |
| Nivel de importancia | Carácter |  | Descripción |
| Alto | A | 70% - 100% | La importancia de la característica y subcaracterísticas son muy alto así que son muy importantes |
| Medio | M | 25% - 69% | La característica y subcaracterísticas no son tan indispensables, pero aun así tiene un buen cargo |
| Bajo | B | 1% - 24% | La necesidad de característica y subcaracterísticas no tienen relevancia |

Nota elaboración: (Elaboración propia)

### 5.1.1 Evaluación de características y subcaracterísticas

|  |  |  |
| --- | --- | --- |
| Características | Subcaracterísticas | Nivel de importancia |
| **Funcionalidad** | Aplicabilidad |  |
| Precisión |  |
| Seguridad |  |
| Conformidad           de       la  funcionalidad |  |
| **Fiabilidad** | Madurez |  |
| Tolerancia a fallos |  |
| Recuperabilidad |  |
| **Eficiencia** | Comportamiento           en       el  tiempo |  |
| **Facilidad de Mantenimiento** | Analizabilidad |  |
| Confiabilidad |  |
| Estabilidad |  |
| Testeabilidad |  |
| Conformidad de facilidad de mantenimiento |  |

Nota elaboración: (Elaboración propia)

https://www.youtube.com/watch?v=Nc4YOEwN6ek

# Diseño de la evaluación

Especificar herramientas para hacer la evaluación

Proceso para desarrolladores

Proceso para evaluadores

Documentación de los módulos de evaluación

# Referencias

Causas de errores de software: <https://www.softwareone.com/es-co/blog/articles/2020/02/17/las-6-causas-mas-comunes-de-la-no-disponibilidad-en-los-sistemas>

Causas de errores de software: <https://www.larepublica.co/internet-economy/cinco-fallas-tecnologicas-mas-comunes-que-pueden-afectar-rendimiento-de-empresas-3082660>

Errores fáltales de la historia: <https://www.fyccorp.com/articulo-10-grandes-errores-de-software>

Ideas principales: <https://repositorio.usmp.edu.pe/bitstream/handle/20.500.12727/1480/baldeon_vej.pdf?sequence=1&isAllowed=y>

Claridad método de evaluación:

<https://www.youtube.com/watch?v=Nc4YOEwN6ek>