Aseguramiento de la calidad de software:
Métricas de calidad

Tarea 1, ISO-9126

Instituto Tecnológico de Costa Rica
Escuela de computación
Ingeniería en computación

Joshua Mata Araya
Adrian López Quesada
Josué Arrieta Salas

20 de agosto de 2016

1. Funcionalidad

Dentro de la métrica de funcionalidad se ha decidido evaluar la calidad del software en aspecto de carácter externo, ya que la satisfacción del cliente es la principal meta en este proyecto, por lo cual es necesario evaluar elementos a conveniencia del mismo.

Los requerimientos establecidos por el cliente pueden ser de carácter importante, pero sin embargo con esta métrica no solo se pretende evaluar la cantidad de requerimientos abarcados, sino la calidad con la que se implementen. De esta manera el cliente puede tener confianza en lo implementado. De aquí el motivo por el cual se escogieron los atributos de idoneidad y exactitud.

La seguridad y fidelidad es un pilar importante, esto debido a que existe una gran confianza de por medio al ingresar los datos de carácter organizacional en el software. Se espera garantizar también al usuario la protección de sus archivos, tanto en el hecho de que los algoritmos no se vean afectados por corrupción de datos, como en el hecho de que sus resultados sean propiedad del usuario únicamente.

1.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Funcionalidad	Seguridad	M
	Idoneidad	M
	Exactitud	H

1.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Seguridad	Qué tan frecuente se corrompen datos en el sistema.	La cantidad de veces que el sistema presenta errores de corrupción de datos entre la cantidad de veces que el usuario ingresa datos es menor o igual a 0.10	-	Selenium y JUnit
Idoneidad	Los requerimientos implementados están de acuerdo a lo implementado	La cantidad de requerimientos implementados entre la cantidad de requerimientos funcionales establecidos es mayor a 0.90, abarcando los más importantes dentro de lo implementado.	-	Selenium
Exactitud	Cantidad de errores en los resultados de los algoritmos	Cantidad de errores entre cantidad de pruebas menor o igual a 0.10	-	JUnit

2. Confiabilidad

Los atributos, de la métrica de confiabilidad,escogidos, son de carácter externo, ya que de ellos depende la visión que los usuario tengan del software, sin embargo la forma de medir esto, es a través de herramientas que no poseen interacción con el usuario.

La importancia de la madurez del software radica en la facilidad que posea para realizar cambio y que el sistema se mantenga estable, para esto es necesario tener pruebas automatizadas que permitan la validación y verificación de cada uno de los componentes del sistema, por lo cual, se pretende medir la madurez con base en la cantidad de requerimientos, tanto funcionales como no funcionales, que hayan sido sometidos a una serie de pruebas automatizadas.

La recuperabilidad fue escogida porque si el software permite pasar a un estado estable, después de algún error grave, en poco tiempo, los usuarios tendrán una buena perspectiva del software, siendo así un aspecto muy valioso a evaluar en este proyecto. Los partidos de fútbol generalmente tienen un lapso de 3 días de diferencia, por lo cual el boca Juniors (Cliente principal) necesita disponer de las funcionalidades del software en momentos muy clave y es de suma importancia garantizar que el software puede ser levantado de nuevo, muy rápido en caso de fallos.

2.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Confiabilidad	Madurez	H
	Tolerancia a fallos	H
	Recuperabilidad	H

2.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Madurez	Cantidad de elementos probados en el software	La cantidad de historias de usuario que presentan un test automatizado entre la cantidad de historias de usuario definidas debe de ser mayor a 0.85	-	JUnit, Selenium
Tolerancia a fallos	Cuantos fallos es capaz el sistema de soportar sin necesidad de caerse por completo	La cantidad de caídas del sistema entre la cantidad de fallos detectados en el sistema debe de ser menor al 0.05	-	Log del sistema
Recuperabilidad	Tiempo que el sistema logra estar sin caerse	La cantidad de horas que el sistema se encuentre caído entre la cantidad de horas que lleva el software en producción debe de ser menor a 0.03	-	Log del sistema

3. Usabilidad

Para la usabilidad se usarán métricas con un enfoque completamente externo. Esto debido a que se considera más importante la perspectiva del usuario respecto al producto de software en ejecución en el tema referente a la usabilidad. Para esta categoría se escogen las métricas: capacidad de ser aprendido, capacidad de ser entendido y capacidad de ser operado.

Para la métrica capacidad de ser aprendido se escoge cómo métrica el tiempo requerido del usuario para usar el programa en su totalidad (subir un video y obtener el video con el análisis automático realizado). Se espera que el programa sea muy fácil de usar. A primera instancia, esta métrica se considera muy buena para tal caso. La junta directiva necesita el análisis automático lo más rápido posible y de esta manera puedan tomar decisiones tácticas de manera eficiente (esta característica se considera la más atractiva del sistema a implementar) y no interrumpir en sus otras actividades.

Para la métrica capacidad de ser entendido se escogió qué tan bien el usuario entiende las entradas y salidas del programa. Para el software a implementación solo habría una salida y una entrada (el vídeo a analizar, y el vídeo analizado). Se escoge esta métrica debido a que se considera sumamente importante que el usuario entienda el propósito del software y se cree que esta métrica es para tal caso.

Para la métrica capacidad de ser operado, se escoge qué tan consistentes son los componentes de interfaz de usuario (en cuanto a funcionalidades y mensajes de interfaz). Es importante que Daniel Angelici reciba una aplicación totalmente congruente y no implementar funcionalidades no requeridas. Hay que cumplir completamente las expectaciones operacionales del usuario y evitar la viscosidad. Se cree que tal métrica es justa para el caso y para encontrar irregularidades dentro de la interfaz de usuario.

3.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Usabilidad	Capacidad de ser entendido	H
	Capacidad para ser aprendido	M
	Capacidad para ser operado	M

3.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Capacidad de ser aprendido (externa)	Tiempo requerido por usuario para usar el programa (de entrada de video a salida de video)	Menos de 160 segundos	-	Java System Timer
Capacidad de ser entendido (externa)	El usuario entiende lo que es requerido cómo entrada al programa, así cómo la salida del mismo	$T = \frac{A}{B} = 1$,donde A es el número de entradas y salidas que el usuario entiende y B es el número total de entradas y salidas del programa	-	Google Forms
Capacidad de ser operado (externa)	Consistencia de los componentes de la interfaz de usuario (funcionalidades y mensajes de interfaz)	$T = 1 - \frac{A}{B} >$ 0,85 ,dónde A es el número de componentes de interfaz considerados inconsistentes y B es el número total de componentes de interfaz	-	Google Forms

4. Eficiencia

Para la externa se usarán métricas con un enfoque tanto interno como externo. Esto debido a que se considera importante la perspectiva del usuario respecto al producto de software en ejecución en el tema referente a la eficiencia, pero también es importante una métrica interna de eficiencia interna para predecir la utilización de recursos de hardware por las computadoras que tendrán el producto de software en un futuro. En este caso los sistemas de la junta directiva del Boca Juniors. Para esta categoría se escogen las métricas: capacidad de comportamiento temporal, utilización de recursos y cumplimiento de eficiencia.

Para la métrica capacidad de comportamiento temporal se escoge el tiempo requerido por frame como métrica externa. Se escoge esta métrica ya que se considera que el procesamiento individual por frame es la unidad más pequeña de trabajo en el software a implementar; y por esta razón la mejor manera para medir tal métrica. Es un buen indicador de eficiencia. Además es externa ya que será visto por la junta directiva. Si el procesamiento por frame es muy lento, el programa en general será lento (la situación opuesta si el procesamiento por frame es rápido).

Para la métrica utilización de recursos se escoge qué tanto porcentaje de memoria está usando el programa como métrica interna. Es interna ya que se usará como predictivo de cómo trabajará en un futuro en las computadoras de la junta directiva del Boca Juniors. Nos da un panorama general temprano. Se escoge tal métrica que ya la utilización de recursos (en este caso memoria) es un aspecto

a tomar en cuenta, ya que si la utilización de recursos es baja, no se afectan otras aplicaciones que los sistemas de la junta directiva tienen que correr y la experiencia de usuario en general mejora.

La métrica cumplimiento de eficiencia no será tomada en cuenta para este proyecto. Esto debido a que no hay especificaciones de normas o convenciones relacionadas con la eficiencia para el sistema a implementar.

4.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Eficiencia	Capacidad de comportamiento temporal	H
	Utilización de recursos	M

4.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Capacidad de comportamiento temporal (externa)	Tiempo de procesamiento requerido por frame	Menos de 20ms	-	Java System Timer
Utlización de recursos (interna)	Porcentaje de memoria usado por el programa	Máximo 5 %	-	Task Manager

5. Mantenibilidad

La mantenibilidad interna se encarga de medir el esfuerzo para realizar cambios o identificar problemas dentro del codigo. Se escogió el aspecto interno en todas las métricas ya que se cree que este tiene mas relevancia cuando se desée modificar el software.

Para la sub-categoria de analizabilidad, se desea medir el indice complejidad cicolomática de los métodos. Este indice se encarga de medir una sección de código y brindar un numero asociado a la cantidad de caminos lineales independientes dentro de él. Una complejidad ciclomática baja no garantiza que el código sea muy sencillo de entender, pero si ayudará a no complicarlo mas de lo que es. Un método con complejidad ciclomatica alta puede simbolizar mal implementacion de este mismo, usando una mala separación de responsabilidades. Metodos con alta complejidad ciclomatica pueden incluir varios controles de flujo que enrealidad oscurecen el código, haciéndolo menos fácil de entender ya sea para nuevas implementaciones o errores espontáneos.

La subcategoria de probabilidad se medira con una metrica de *Code Coverage* la cual se traduciria a cobertura de codigo. Se desea saber la cobertura del codigo con respecto a las pruebas, para asi poder minimizar esfuerzos manuales para atender errores encontrados. Facilitar a los desarrolladores a darles mantenimiento al codigo, donde pueden modificar o agregar alguna funcionalidad nueva y puede garantizar que esta no daño otra.

Por último, la subcategoria de la cambiabilidad. Esta subcategoria es de suma importancia, ya que especifica los esfuerzos para modificar el software. Se utilizará el indice de mantenibilidad. Este indice utiliza la siguiente formula:

$$171 - 25\ln(HV) - 0{,}23CC - 16{,}2\ln(LOC) + 50{,}0\sin(\sqrt[2]{(2{,}46 * COM)}) \quad (1)$$

Con HV: Volumen de Halstead, número de operaciones por el logaritmo base dos de operaciones distintas. CC: complejidad ciclomática. LOC: lineas de código. COM: cantidad de comentarios. Esta fórmula da un resultado entre 0 y 100, entre mayor sea el resultado, mas *mantenible* es el proyecto. Esta métrica es altamente utilizada en el entorno de .NET.

5.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Mantenibilidad	Capacidad para ser analizado	M
	Capacidad para ser probado	H
	Capacidad de ser cambiado	H

5.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Capacidad para ser analizado (interna)	Indice de complejidad ciclomatica	Igual o por debajo de 10	-	JHawk5 Java Metrics
Capacidad para ser probado (interna)	Cobertura de las pruebas por el codigo (Code Coverege)	Al menos 80 %	-	EclEmma
Capacidad de ser cambiado (interna)	Indice de mantenibilidad	Al menos un 80 %	-	JHawk5 Java Metrics

6. Portabilidad

Para la categoría de portabilidad, se decidió tomarlo desde una perspectiva externa. Aunque se vaya diseñar de manera en que sea portable, lo esencial es que su funcionamiento sea el mas optimo en cualquier dispositivo probado por el usuario.

Con respecto a la subcategoria de la adaptabilidad, se desea utilizar una métrica que mida el software adaptado a su entorno. Debido a que es una funcionalidad web, lo común es que la gente utilice diferentes tipos de navegadores, por lo que se espera no limitar al usuario respecto a su elección de navegador.Se espera que al menos 3 navegadores cumplan este objetivo. Relacionado al entorno en donde se planea utilizar el software. Lo mas recomendable sería probar su adaptabilidad en el sistema que los árbitros planean utilizarlo.

La métrica utilizada para la instabilidad es externa pero no esta totalmente relacionado a los usuarios finales, debido a que su funcionalidad es por web no se debe realizar ningún tipo de instalación.Sin embargo, para esta primera fase debe existir un servidor que administre la lógica del proyecto, por lo que se consideró el tiempo de instalación del software relacionado al servidor. Se especificara todo lo necesario para poner a funcionar la aplicación de manera local en el sistema operativo de Windows, el usuario deberá utilizar la documentación proporcionada para iniciar y finalizar la instalación del software. En una siguiente entrega, se agregará la prueba de instalación de manera local en los dispositivos utilizados por los árbitros o entrenadores que desean tener acceso a este software sin necesidad de tener acceso a web.

Finalmente, se desea que la capacidad de coexistencia sea bastante alta en este proyecto. Como primera fase, se desea que la funcionalidad web no genere problemas con otros procesos, tomando en cuenta el cliente que desea utilizarlo. Además que se requiere que no hayan problemas de recursos en el entorno donde se establecerá el servidor con la lógica, para evitar posibles problemas que denieguen el acceso web a este mismo. A futuro, la coexistencia con otros procesos es esencial debido a su utilización en los dispositivos de los árbitros o entrenadores. No se desea que hayan problemas al compartir recursos de manera local, por lo que en otra fase se realizaran pruebas para estos dispositivos en específico.

6.1. Atributos y su prioridad

Característica	Sub-Característica	Peso
Portabilidad	Atributo de Adaptabilidad	M
	Atributo de Instalabilidad	L
	Capacidad de coexistencia	H

6.2. Métricas usadas

Característica	Métrica	Nivel Requerido	Resultado Actual	Herramienta
Atributo de Adaptabilidad (externa)	Sistema de Software adaptable al entorno	Al menos 3 navegadores diferentes con aprobación de $X \geqslant 90\,\%$	-	Manual, X= 1-$\frac{A}{B}$ donde A: prueba funcional fracasada, B: cantidad de pruebas funcionales realizadas
Atributo de Instalabilidad (externa)	Tiempo de instalación desde cero sin ayuda externa	$X \geqslant 75\,\%$ Donde el tiempo máximo de instalación sera de 45 minutos.	-	Manual, X = 1-$\frac{A}{B}$.A: tiempo requerido para la instalación, sin contar tiempos de descarga. B: Tiempo máximo de instalación.
Capacidad de coexistencia (externa)	Coexistencia disponible	$X \geqslant 90\,\%$, $90\,\%$ de disponibilidad.En el transcurso de 3 horas.	-	X = $\frac{A}{B}$ donde A: cantidad de errores encontrados, B: tiempo transcurrido de funcionamiento. Se utilizara los logs del software y detección de errores del Sistema Operativo para monitorear errores de consistencia