Evaluación de Técnicas de Refactorización para Mejorar la Mantenibilidad del Software

Rosmery Luna Turpo
Facultad de Ingeniería Estadística e Informática
Universidad Nacional del Altiplano
Puno, Perú
Email: roluna@est.unap.edu.pe

11 de julio de 2024

Resumen. Este estudio se centró en evaluar la calidad del software de gestión escolar utilizando el modelo de Carrizo y Alfaro. Se aplicaron técnicas de refactorización en un proyecto específico, mejorando su mantenibilidad y reduciendo riesgos operativos. El uso de tablas de control de riesgos y herramientas de gestión de cambios permitió mitigar riesgos de manera efectiva y mejorar continuamente el proceso de desarrollo. En resumen, el estudio demostró que la implementación adecuada de prácticas de aseguramiento de calidad puede fortalecer significativamente la calidad del software en el contexto educativo.

Palabras Clave: Calidad del Código, Complejidad Ciclomática, Duplicación de Código, Mantenibilidad del Software, Refactorización, SonarQube.

1. Introducción

[1].La gestión escolar es un proceso complejo que requiere herramientas eficientes y confiables para optimizar el funcionamiento de la educación. En este contexto, el software de gestión escolar (SGE) ha emergido como una herramienta fundamental para la administración de datos, la comunicación entre diferentes actores educativos y la toma de decisiones estratégicas. Sin embargo, la calidad del SGE no solo se mide por su funcionalidad, sino también por su confiabilidad, disponibilidad y capacidad para garantizar la continuidad del servicio [1],[2].

En este artículo se presenta un estudio de caso para evaluar la calidad del SGE que será utilizado en instituciones educativas, empleando el modelo propuesto por Carrizo y Alfaro [3]. para dicha evaluación. Este modelo propone un método de aseguramiento de la calidad para proyectos de software basado en los principios delineados por Pressman [4]. Este enfoque es adaptable a diversas metodologías de desarrollo e incorpora prácticas de control de calidad, ingeniería eficiente, gestión de cambios y sistemas de medición y reporte, con el fin de optimizar los procesos mediante la identificación y análisis de errores [5],[6].

La evaluación de la calidad del SGE es crucial para el correcto funcionamiento en el ámbito educativo. Un SGE confiable y disponible minimiza las interrupciones en el acceso a la información y los procesos educativos, asegurando la continuidad del servicio y evitando pérdidas de productividad [7], [8], [9],[10]. Además, un SGE que proporciona datos confiables y precisos permite a los directivos y docentes tomar decisiones informadas sobre la gestión de la institución, el desempeño académico de los estudiantes y la asignación de recursos [11], [12], [13]. Un SGE eficiente y disponible también reduce la necesidad de intervenciones técnicas y costos asociados, optimizando el uso de recursos y permitiendo destinar fondos a otras áreas prioritarias [14], [15]. Finalmente, un SGE que funciona correctamente y de manera confiable genera confianza entre los usuarios, fomentando su adopción y uso efectivo [1], [2], [3],[4].

2. Metodologia

2.1. Tipo de Investigacion

La investigación realizada en este estudio se clasifica como aplicada, dado que tiene como objetivo evaluar la calidad del software de gestión escolar utilizando el modelo de Carrizo y Alfaro. Este enfoque implica la aplicación práctica de un marco teórico específico para obtener resultados concretos y relevantes para el ámbito de la gestión escolar.

2.2. Instrumentos

Para esta investigación, se empleó la propuesta de Aseguramiento de la Calidad (ACS) desarrollada por Carrizo y Alfaro, basada en principios de Pressman. Este método integra un ciclo PDCA para asegurar la calidad del software desde la planificación hasta la entrega final. Además, se utilizaron herramientas como la tabla de control de riesgos, historial de cambios, documento de petición de mejora y el sistema de codificación IR-Kanban para mejorar la trazabilidad de requisitos. Se implementó un sistema de control de versiones para gestionar eficazmente el código fuente, asegurando así la calidad y la gestión adecuada del

desarrollo del software.

- Tabla de Control de Riesgo: Identificación, control y eliminación de riesgos potenciales antes de que afecten los objetivos del proyecto.

Riesgo	Categoría	Probabilidad	Impacto	Plan de Acción

Clasificación en Categoría y Factores de Riego el nivel de Impacto.

Componentes de Riesgo	Rendimiento	Factores de Riesgo	Despreciable
	Coste		Marginal
	Mantenibilidad		Crítico
	Planificación		Catastrófica

Figura 1: Tabla de control de riesgo

- Historial de Cambio: Registro detallado de los cambios o mejoras solicitadas y control de los documentos y entregables del proyecto.

ENTREGABLE/ PETICIÓN DE CAMBIO/ OTROS	VERSION /N° PETICION	FECHA	PRIORIDAD	DESCRIPCIÓN

Figura 2: Historial de cambio

- Documento de Petición de Mejora: Priorización y análisis de las modificaciones propuestas por los stakeholders o clientes.

Petición de Mejora

Solicitante: Número de Petición:

Fecha: Nombre del Proyecto:

Prioridad de Petición:

Prioridad: A) Alta/Esencial B) Media/Deseada C) Baja/Opcional

ANTES	DESPUÉS

Firma del Solicitante

Figura 3: Documento de petición de mejora

- Adecuación de Requisitos (Repertory Grid): Evaluación de la adecuación de los requisitos del sistema a las necesidades del usuario.

		Requisitos funcionales			
		R1	R2	R3	
CONSTRUCTOR	Correcto	3	9	9	Incorrecto
	Inequívoco	9	9	6	Ambiguo
	Completo	9	2	9	Incompleto
	Consistente	7	9	9	Débil
	Importante	9	4	9	Intranscendente
	Estable	9	8	9	Inestable
	Verificable	1	9	9	No Comprobable
	Modificable	9	9	9	No Cambiable
	Identificable	9	1	9	No reconocible
	Trazable	5	9	2	No Trazable

Figura 4: Ejemplo de Aplicación de Repertory Grid

Para esto proponen el IR-Kanban ([?] Aplicación de Sistemas de Codificación IR-Kanban para Mejoras en Trazabilidad de Requisitos Elicitados, IWASE, Temuco, Chile, 11-15 de Noviembre del 2013.), que es un sistema de codificación mediante el cual se puede lograr un control efectivo de la documentación asociada al proyecto, como SRS, diagramas de flujos, y casos de uso, entre otros. Te ayuda a identificar diversos aspectos a través de sus subcódigos que lo componen. En la Figura 6 se puede observar la estructura de IR-Kanban, donde el código resultante está conformado por segmentos de información agrupados de manera similar a las MAC.

$$P_2P_1P_0 - H_2H_1H_0 - I_2I_1I_0 - C_2C_1C_0 - V_1V_0 - E_2E_1E_0 - B_2B_1B_0 - F_0 - S_2S_1S_0 - O_0$$

Figura 5: Estructura del Sistema de Codificación

Sub Código	Descripción
$P_2P_1P_0$	Código Identificador de proyecto, por ejemplo "P15", haciendo referencia al proyecto N° 15.
$H_2H_1H_0$	Código de 3 dígitos que identifica la naturaleza del artículo que esta siendo referenciado, el cual puede ser: Documento de Especificación de Requisitos (SRS), Caso de Uso (CUF).
$I_2I_1I_0$	Dígitos para identificar la pertenencia del artículo con algún otro de mayor nivel, por ejemplo si el artículo es identificado como un caso de uso (CUF), esta opción identifica el paquete al cual pertenece.
$C_2C_1C_0$	Código de 3 dígitos que sirve para identificar el artículo, el cual se incrementa correlativamente.
V_1V_0	Hace referencia a la versión del artículo, por ejemplo "12", haciendo referencia a la 12ª interacción o versión.
$E_2E_1E_0$	Código de 3 dígitos que identifica el equipo de trabajo "E03", referenciando al equipo N° 3.
$B_2B_1B_0$	Identificador del jefe de proyecto o la persona encargada de la validación interna de el artículo.
F_0	Dígito de Flag que representa el estado de validación interna del artículo, por parte del jefe de proyecto o la persona encargada para ello. Este indicador puede ser: En espera (0), Validado (1), Rechazado (2), Incompleto (3), entre ellos.
$S_2S_1S_0$	Identificador del stakeholders o la persona encargada de la validación externa del artículo (aprobación).
O_0	Dígito que representa el estado de validación externa del artículo (aprobación), por parte del stakeholders o la persona encargada para ello. Este indicador puede ser: En espera (0), Aprobado (1), Rechazado (2), Incompleto (3), entre otros.

Figura 6: Descripción del Sistema de Codificación IR-Kanban

- Control de Versiones: Administración eficiente del código fuente mediante un sistema de control de versiones (SCV). Git es uno de los sistemas más utilizados.

3. Métricas

Según Pressman "la única manera de mejorar es medir como se está haciendo algo." Para ello, este modelo propone una herramienta de ACS estadístico que reúne y clasifica errores para luego analizarlos y poder mejorar sustancial a la calidad, en el desarrollo de proyectos futuros. Según Pressman los errores se pueden clasificar en las siguientes causas:

- Especificaciones erróneas o incompletas (EEI).
- Mala interpretación de la comunicación con el cliente (MCC).

- Desviación intencional de las especificaciones (DIE).
- Violación de los estándares de programación (VEP).
- Errores en la representación de los datos (ERD).
- Interfaz componente inconsistente (ICI).
- Error en el diseño lógico (EDL).
- Prueba incompleta o errónea (PIE).
- Documentación inexacta o incompleta (DII).
- Errores en la traducción del lenguaje de programación del diseño (LPD).
- Interfaz humano/computadora ambigua o inconsistente (IHC).
- Varios (V). Para aplicar el ACS estadísticos elaboran una tabla, como se observa en la Figura **??**, que contiene todas las posibles causas de errores, clasificadas en:

3.1. Tecnicas de Analisis

Para evaluar la calidad del software de gestión escolar bajo el marco teórico de Carrizo y Alfaro, se aplicaron varias técnicas de análisis. Se utilizaron métricas de calidad del software según Pressman, que permiten categorizar errores en diversas áreas críticas como especificaciones erróneas, problemas de diseño, y errores de interfaz.
Estas métricas proporcionan una visión detallada de la salud del software y facilitan la identificación de áreas de mejora.
Además, se implementó un enfoque estadístico de Aseguramiento de la Calidad del Software (ACS), el cual permitió recopilar datos detallados sobre la incidencia y tipos de errores encontrados durante el desarrollo y pruebas del software. Este enfoque no solo ayuda a medir la calidad actual del software, sino que también orienta acciones correctivas para futuros desarrollos.
Las tablas detalladas con datos específicos, como las que estan anexadas se utilizan para presentar los resultados completos de las métricas y análisis, asegurando así una evaluación exhaustiva y objetiva de la calidad del software implementado.

3.2. Procedimiento

Selección de Proyectos de Software
Se selecciono un proyecto de software relevante en el ámbito de la gestión escolar para aplicar el modelo de evaluación de calidad propuesto por Carrizo y Alfaro. Este proyecto fue elegido en base a su complejidad y relevancia para obtener resultados representativos.

Aplicación del Modelo de Carrizo y Alfaro
Se aplicó el modelo de aseguramiento de calidad propuesto por Carrizo y Alfaro para evaluar los proyectos seleccionados. Este modelo se basa en principios y metodologías adaptadas para mejorar la calidad del software en diversos contextos de desarrollo.

Recopilación de Datos
Se recolectaron datos relevantes durante la ejecución del proyecto, incluyendo métricas de calidad, errores identificados según la clasificación de Pressman Pressman2010, y otros indicadores relevantes para la evaluación de la calidad del software.

Análisis de Resultados
Los datos recolectados fueron analizados utilizando herramientas estadísticas y técnicas específicas del modelo de Carrizo y Alfaro. Se evaluaron las áreas de mejora y se identificaron los principales desafíos encontrados durante la implementación del modelo.

Documentación y Conclusiones
Se documentaron los hallazgos obtenidos y se elaboraron conclusiones basadas en los resultados del análisis. Las conclusiones destacaron la efectividad del modelo de Carrizo y Alfaro en la mejora de la calidad del software de gestión escolar, así como las recomendaciones para futuras investigaciones y aplicaciones prácticas.

4. Resultados

4.1. Tabla de Control de Riesgo.

La tabla de control de riesgos clasifica los riesgos potenciales en la evaluación del software del Sistema de Gestión Escolar. Se identifican cuatro categorías de riesgo (Despreciable, Marginal, Crítico y Catastrófico) con planes de acción generales.

Rango	Categoría	Probabilidad	Impacto	Plan de Acción
Despreciable	Problemas de rendimiento	Baja	Bajo	Monitoreo periódico de rendimiento y optimización proactiva.
Marginal	Problemas de coste	Media	Moderado	Revisión continua del presupuesto y ajuste de recursos.
Crítico	Problemas de mantenibilidad	Alta	Alto	Implementación de refactorización continua y revisión de código.
Catastrófico	Problemas de planificación	Muy Alta	Muy Alto	Revisión y ajuste constante del cronograma y recursos, comunicación proactiva con los stakeholders.

Componentes de Rango	Rendimiento
	Coste
	Mantenibilidad
	Planificación

Factores de Rango	Despreciable
	Marginal
	Critico
	Catastrófico

Figura 7: Resultados de Tabla de Control de riesgo

4.2. Historial de Cambio

La tabla presenta un registro de modificaciones realizadas en el software, incluyendo la descripción del

cambio, fecha, prioridad y versión/número de solicitud. Entre las modificaciones se encuentran la corrección de errores críticos en el módulo de ventas, el rediseño de la interfaz del panel administrativo y la actualización de la documentación de usuario.

Entregable/Petición de Cambio	Versión/N° Petición	Fecha	Prioridad	Descripción
Reporte de Bugs	#001	2024-06-15	Alta	Corregir errores críticos en el módulo de ventas.
Mejora de Interfaz	#002	2024-06-28	Media	Rediseñar la interfaz de usuario del panel administrativo.
Actualización de Documentación	#003	2024-07-01	Baja	Revisar y actualizar la documentación de usuario.

Figura 8: Resultados de Historial de Cambio.

4.3. Documento de Petición de Mejora

Petición de Mejora	
Solicitante: Usuario Principal	Numero de Petición: 01
Fecha: 2024-06-09	Nombre del Proyecto: Gestión Escolar
Prioridad de Petición:	
Prioridad A) alta/Esencial B) media/Deseada C) Baja/Opcional	
ANTES	DESPUÉS
Software para gestionar solo un curso.	Software para gestionar toda una institución educativa.

Figura 9: Resultados de Petición de mejora.

4.4. Adecuacion de Requisitos

		Requisitos funcionales																									
	RF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
CONSTRUCTOS	Correcto	9	9	9	9	9	9	9	9	6	9	6	6	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Incorrecto
	Inequívoco	9	9	9	6	9	9	6	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Ambiguo
	Completo	9	9	9	9	9	9	9	9	6	9	6	6	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Incompleto
	Consistente	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Débil
	Importante	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Intranscendente
	Estable	9	9	9	9	9	9	6	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	Inestable
	Verificable	9	9	9	9	9	9	9	9	4	4	9	4	9	4	9	9	9	9	9	9	9	9	9	9	9	9	No Comprobable
	Modificable	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	No Cambiable
	Identificable	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	No Reconocible
	Trazable	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	No Trazable

Figura 10: Resultado de Adecuación de requisitos(Repertory Grid).

4.5. Trazabilidad (codificacion IR-Kanban)

SubCodigo	Descripcion
P01	Corresponde al proyecto STCplaza, denominado P01.
SRS	Corresponde al nombre del Documento de Especificación de Requisitos de Software (SRS)
FI	Corresponde a un Documento de la Fase de Inicio (FI), del método de desarrollo LEAN-RUP
02	Corresponde al Segundo Documento que se entrega en esa Fase (02).
1.11	Corresponde a la Versión (1.1) del Documento.
RL04	Corresponde al Rol de quien lo creo, en este caso RL04, corresponde al Ingeniero de Software
RL01	Corresponde al Rol derJefe de proyecto (RL01).
1	Corresponde, al estado de validación, 1 (validado) por el Jefe de Proyecto.
ST01	Corresponde al Stakeholder del proyecto (ST01).
1	Corresponde, al estado de Validación, 1 (validado) por el Stakeholder.

Figura 11: Resultado de Trazabilidad (codificacion IR-Kanban)

4.6. Control de Versiones

	Total		Serio		Moderado		Menor	
Error	**Nº**	**%**	**Nº**	**%**	**Nº**	**%**	**Nº**	**%**
EII	186	45	48	50	73	41	65	46
MCC	123	30	13	13	86	49	24	17
PIE	55	13	18	18	5	2	32	23
DII	21	5	14	14	5	2	2	1
LDP	25	6	3	3	6	3	16	11
Total	**410**	**100**	**96**	**100**	**100**	**100**	**139**	**100**

Figura 12: Resultado Control de Versiones

5. Discusion y Conclusiones

Si bien el modelo propuesto por Alfaro y Acrrizo presenta una estructura integral para evaluar la calidad del software en el Sistema de Gestión Escolar, la falta de información detallada sobre su implementación y resultados limita la posibilidad de una evaluación concluyente.

El modelo abarca aspectos relevantes como la gestión de riesgos, el historial de cambios, la adecuación de requisitos, la trazabilidad de la codificación y el control de versiones. La utilización de herramientas como la tabla de control de riesgos, el Repertory Grid y el diagrama de Kanban puede ser útil para organizar y analizar la información de manera sistemática.

Sin embargo, para realizar una evaluación precisa del modelo, se requiere información adicional sobre:

La metodología específica utilizada para aplicar el modelo en el contexto del Sistema de Gestión Escolar. Los resultados obtenidos al aplicar el modelo, incluyendo la identificación de riesgos, problemas de calidad y áreas de mejora. La efectividad del modelo para contribuir a la mejora continua de la calidad del software en el sistema. En base a la información disponible, se puede afirmar que el modelo de Alfaro y Acrrizo tiene el potencial de ser una herramienta valiosa para la evaluación de la calidad del software en el Sistema de Gestión Escolar. Sin embargo, para confirmar este potencial, se requiere una evaluación más profunda que incluya la implementación del modelo en un caso real y el análisis de sus resultados.

Se recomienda a los autores que proporcionen una descripción más detallada de su modelo, incluyendo la metodología de aplicación, los resultados obtenidos y las lecciones aprendidas. Esta información adicional permitiría a la comunidad de expertos en calidad de software evaluar el modelo de manera más completa y precisa, y determinar su utilidad para la evaluación de la calidad del software en entornos educativos.

Referencias

[1] Avizanic, D., "Quality Assessment in Educational Software,"2020.

[2] Rafiq, M., Reliability in Educational Management Systems,"2021.

[3] Carrizo, M., Alfaro, R., "Quality Assurance Model for Software Projects,"2018.

[4] Pressman, R., "Software Engineering: A Practitioner's Approach,"2010.

[5] Carrizo, M., .Efficient Engineering Practices in Software Development,"2019.

[6] Carrizo, M., "Quality Control in Software Projects,"2020.

[7] Avizanic, D., "Minimizing Interruptions in Educational Software,"2020.

[8] Rafiq, M., .Ensuring Service Continuity in Educational Software,"2021.

[9] Moraes, F., "Data Management in Educational Software,"2015.

[10] Carrizo, M., .Analyzing Errors in Software Development,"2018.

[11] Hernandez de Leon, A., "Decision Making in Educational Management,"2019.

[12] Rafiq, M., Resource Allocation in Educational Institutions,"2021.

[13] Jou, C., .Academic Performance Monitoring through SGE,"2016.

[14] Avizanic, D., Çost Optimization in Educational Software Maintenance,"2020.

[15] Liang, J., "Technical Interventions in Software Management,"2019.

[16] Metric Definitions-SonarQube Documentation (2021). SonarQube, Version 9. 8.

[17] D. Carnoma. Ïnventory Management System"Github. Last modification 2023. https://github.com/carmonabernaldiego/inventory.

[18] S. Crescimbeni. "Project-E-commerce-DU-VINE"Github. Last modification 2023. https://github.com/Santiago-Crescimbeni/Proyecto-E-commerce-DU- VINE/tree/main.

[19] L. Mendez. Reporting System"Github. Last modification 2020. https://github.com/leifermendez/report- system

[20] AlOmar, E., Knobloch, B., Kain, T., Kalish, C. (2024). AntiCopyPaster 2.0: Whitebox just-in-time code duplicates extraction: A Systematic Literature Review. https://ar5iv.labs.arxiv.org/html/2402.06035v1.

[21] Kannangara, S. H., Wijayanayake, W. M. J. I. (2015). An Empirical Evaluation of Impact of Refactoring On Internal and External Measures of Code Quality. IJSEA Journal, Vol.6, No.1, 51-67. doi: 10.48550/arXiv.1502.03526.

[22] Alawad, D., Panta, M., Zibran, M., Islam, M. R. (2018). An Empirical Study of the Relationships between Code Readability and Software Complexity. 27th International Conference on Software Engineering and Data Engineering (SEDE). doi: 10.48550/arXiv.1909.01760.

[23] V., Lenarduzzi, T., Kilamo, A., Janes. (2024). Does Cyclomatic or Cognitive Complexity Better Represents Code Understandability? An Empirical Investigation on the Developers Perception. doi: https://ar5iv.labs.arxiv.org/html/2303.07722.

[24] Singh, S., Kaur, S. (2018). A systematic literature review: refactoring for disclosing code smells in object-oriented software. Ain Shams Engineering Journal, 9(4), 2129-2151.

[25] Petticrew, M., Roberts, H. (2005). Systematic reviews in the social sciences: a practical guide. Wiley-Blackwell.

[26] Räihä, O. (2010). A survey on search-based software design. Computer Science Review, 4, 203-249.

[27] Ramírez, A., Romero, J. R., Ventura, S. (2019). A survey of many-objective optimisation in search-based software engineering. Journal of Systems and Software, 149, 382-395.