Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), marzo-abril 2024,
Volumen 8, Número 2.

https://doi.org/10.37811/cl_rcm.v8i2

# MODELO DE REINGENIERÍA DE SOFTWARE PARA SISTEMAS LEGADOS DE UNA INSTITUCIÓN DE EDUCACIÓN SUPERIOR EN MÉXICO

## SOFTWARE REENGINEERING MODEL FOR LEGACY SYSTEMS OF A HIGHER EDUCATION INSTITUTION IN MÉXICO

**Cuautli David García López**
Tecnológico Nacional de México, México

**Samuel Efrén Viñas Álvarez**
Tecnológico Nacional de México, México

**Dario Dávalos Hernández**
Tecnológico Nacional de México, México

**DOI:** https://doi.org/10.37811/cl_rcm.v8i2.10559

# Modelo de Reingeniería de Software para Sistemas Legados de una Institución de Educación Superior en México

**Cuautli David García López**[1]
M21650560@zitacuaro.tecnm.mx
https://orcid.org/0000-0002-9889-3753
Tecnológico Nacional de México
Campus Zitácuaro
México

**Samuel Efrén Viñas Álvarez**
samuel.va@zitacuaro.tecnm.mx
https://orcid.org/0000-0001-5891-2801
Tecnológico Nacional de México
Campus Zitácuaro
México

**Dario Dávalos Hernández**
dario.dh@zitacuaro.tecnm.mx
https://orcid.org/0009-0009-2222-8595
Tecnológico Nacional de México
Campus Zitácuaro
México

**RESUMEN**

El presente trabajo presenta una investigación realizada en el Tecnológico Nacional de México campus Zitácuaro cuyo objetivo fue desarrollar y aplicar un modelo de reingeniería de software con la finalidad de mejorar la productividad y eficiencia de sistemas legados en esa institución. La investigación se basó en un estudio del estado del arte y una propuesta metodológica aplicada al caso de estudio. El modelo propone cuatro etapas: evaluación del sistema, desarrollo ágil, abstracción y aplicación, como resultados tangibles se obtuvieron instrumentos agilizaron las etapas del modelo, optimizaron el proceso de reingeniería de software, la reutilización de componentes de software, así como la gestión de sistemas legados.

***Palabras clave:*** modelo, metodología, reingeniería, software, sistema legado

[1] Autor principal.
Correspondencia: samuel.va@zitacuaro.tecnm.mx

# Software Reengineering Model for Legacy Systems of a Higher Education Institution in México


## ABSTRACT

The present work presents a research carried out at the Tecnológico Nacional de México Zitácuaro campus whose objective was to develop and apply a software reengineering model with the purpose of improving the productivity and efficiency of legacy systems at that institution. The research was based on a study of the state of the art and a methodological proposal applied to the case study. The model proposes four stages: system evaluation, agile development, abstraction and application, as tangible results, instruments were obtained that streamlined the stages of the model, optimized the software reengineering process, the reuse of software components, as well as systems management. legacies.



***Keywords*:** model, methodology, reengineering, software, legacy system

*Artículo recibido 20 febrero 2024*
*Aceptado para publicación: 25 marzo 2024*

## INTRODUCCIÓN

De acuerdo con Sommerville I. (2014) la Reingeniería de Software (RS) es el proceso que se aplica a sistemas informáticos que presentan problemas de obsolescencia o ausencia de características de calidad de software, pero que siguen cumpliendo con los objetivos para los cuales fueron creados y que requieren de mejoras para atender nuevas necesidades, al aplicar la reingeniería se debe volver a revisar la documentación correspondiente, revisar la arquitectura del sistema para reorganizar la estructura y volver a implementarla en tecnología o lenguaje de programación moderno que satisfaga las necesidades actuales y del mercado, la RS se aplica en casos cuando:

- El código no tiene una estructura clara o lógica y puede que no exista ningún tipo de documentación.
- Cuando el soporte de hardware y software en los sistemas actuales queda obsoleto debido a cambios en las políticas de la organización, la competitividad del mercado o las necesidades para mantener el sistema.
- Cuando los desarrolladores de los sistemas legados no están disponibles para verificar o explicar la información y la única fuente es el código de software actual.
- Cuando los sistemas legados, se vuelven difíciles y costosos de cambiar.

En este documento se presentan los resultados de una investigación que tuvo como objetivo principal diseñar y aplicar una guía para la aplicación de reingeniería de software de sistemas legados en el Tecnológico Nacional de México campus Zitácuaro, a partir de esta guía se desarrolla un modelo que permite mantener métodos de reingeniería personalizados.

Actualmente en dicha institución se cuenta con un equipo de tres programadores para el desarrollo de software para sistematizar los procesos administrativos de la institución, por lo que el equipo de desarrollo se enfrentó al reto de renovar los sistemas legados existentes mediante un modelo que se ajuste al entorno de trabajo tomando como base los modelos ya existentes para la RS.

La razón que motivó esta investigación surgió a partir de la necesidad de la mejora del desarrollo y mantenimiento del software ya que es la base para convertir un sistema legado en un sistema nuevo, además de contribuir al campo de estudio de RS con la finalidad de que la guía propuesta pueda contribuir en casos de estudio similares. El uso del modelo propuesto en este trabajo permitirá la reconstrucción y evolución de los sistemas, asimismo para el caso de estudio permitirá que nuevo

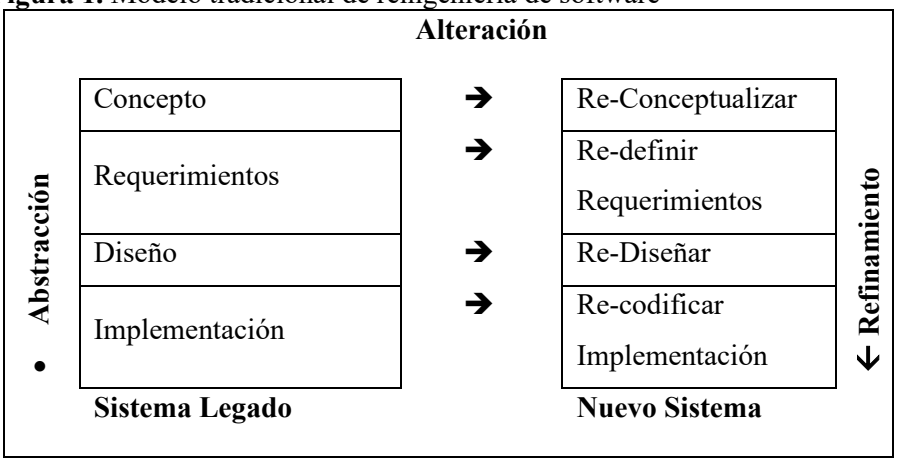
personal que se integre al equipo de trabajo pueda comprender con funcionan los sistemas actuales con la finalidad de realizar el mantenimiento que sea necesario, como fue el caso de estudio presentado por Bianchiotti F., Casas S. (2014) en el que se enfrentaron a la experiencia de reingeniería de un sistema legado y a partir de ahí obtuvieron como resultado una guía para casos similares.

La importancia de contar con un modelo de RS es porqué en la actualidad un proceso de negocio y los sistemas de software afecta fuertemente el desempeño de la ejecución del proceso de negocio. Se necesitan metodologías y herramientas para mantener un proceso de negocio alineado con los sistemas de software de soporte incluso cuando evolucionan (Aversano L., Grasso C., and Tortorella M., 2016). Barbieri S. (2014) sugiere que tanto en el desarrollo de proyectos de software, mantenimiento y reingeniería se usen marcos de trabajo especificando criterios que identifiquen cuando los sistemas legados son aptos para invertir tiempo en ellos, para ello sugiere una revisión de algunos modelos como PSP, RUP, XP y CMMI; por otra parte Baldoneo J.A. (2017) menciona que CMMI y la metodologías agiles si se utilizan correctamente pueden ayudar a las organizaciones a que sus proyectos alcancen un equilibrio optimo, además de sugerir el uso de CMMI para optimizar los procesos de una organización. El modelo de maduración CMMI ofrece tres principios para la gestión de proyectos cuando se desea usar RS, objetivos claros, mejora continua y toma de decisiones basada en datos (Rodríguez M. V. 2017); estos principios se consideraron en la propuesta del modelo.

Para desarrollar el modelo propuesto se consideró lo que actualmente se tiene del sistema legado en cuanto a compontes, documentación e ingeniería de software como sugiere Medina A.S., Chaparro M. L (2013), Seacord, R. C., Plakosh, D., & Lewis, G. A. (2003) y Gómez S.R. and Moraleda E. (2020), además se realizó una revisión de los antecedentes de RS con fin de considerar sus aportaciones al campo de estudio y a la presente investigación.

Majthoub M.,Qutqut M. H. and Odeh Y. (2018) en su artículo titulado: “Software Re-engineering: An Overview”, hacen mención del modelo tradicional al desarrollar RS el cual está basado en tres principios, en la abstracción, alteración y refinamiento; la abstracción permite simplificar la información del sistema para obtener lo más importante, en cuanto la alteración consiste en eliminar o modificar con respecto a la funcionalidad del sistema y el refinamiento consiste en realizar información más detallada del sistema, a continuación se puede visualizar el modelo, véase Figura 1:

**Figura 1.** Modelo tradicional de reingeniería de software




De acuerdo con Majthoub M.,Qutqut M. H. and Odeh Y. (2018) para poder llevar a cabo el modelo de la Figura 1, se puede utilizar Ingeniería Inversa la cual consiste en analizar un sistema para identificar los componentes del sistema y sus interrelaciones y crear representaciones del sistema en otra forma o un nivel superior de abstracción, el objetivo es recuperar información perdida y aplicar la reutilización; normalmente la ingeniería inversa precede a la reingeniería. Para aplicar ingeniería inversa se puede aplicar el enfoque de Bin Bang, incremental y evolutivo, los cuales permiten brindar apoyo al iniciar con la reingeniería.

Otro estudio sobre reingeniería en el proceso de desarrollo de software es el de Bhavsar K., Shah V., Gopalan S. (2019), en su propuesta integran Machine Learning como alternativa para alcanzar un nivel deseado en la calidad del software a través de la evaluación comparativa de los procesos existentes.

Stojkov A., Kazi Lj., and Blazic M. (2020) mencionan en su investigación que al realizar reingeniería de software es importante incluir el uso de documentación en el código fuente para reconstruir y reorganizar un sistema existente y aplicar ingeniería inversa con el uso de modelos y fuentes adecuadas para favorecer la reconstrucción del sistema; en esta investigación al aplicar RS se basaron en SOLID; SOLID es un acrónimo compuesto de 5 principios para la codificación y el diseño orientado a objetos, que en forma general establece las bases para la codificación y comportamiento de clases, objetos y funcionamiento de los módulos de un sistema.

Singh J., Dhindsa K. S. and Singh J. (2020) en su estudio aplicaron el marco de trabajo SCRUM el cual permitió reducir la complejidad al realizar RS, para ello se basaron en una tabla de tareas de reingeniería usando SCRUM, véase Figura 2.

**Figura 2.** Tareas de Reingeniería usando SCRUM

| No. | Sprint | Tareas | Sub tareas |
|---|---|---|---|
| 1 | Primera semana | Ingeniería inversa (abstracción) | Entender la estructura existente |
| | | | Análisis y diseño de especificaciones |
| | | | Análisis de requerimientos |
| | | | Retrospectiva |
| 2 | Segunda semana | Alteraciones o cambios | Análisis y diseño de clases |
| | | | Reconstrucción del comportamiento de clases |
| | | | Pruebas |
| | | | Retrospectiva |
| 3 | Tercera semana | Reingeniería hacia delante (refinamiento) | Integración |
| | | | Pruebas de integración |
| | | | Retrospectiva |

SCRUM como una metodología ágil la cual ha comprobado que al dividir el problema a resolver en fragmentos pequeños llamados Sprints se puede generar un mayor avance en el desarrollo, evitando la saturación de trabajo, además de adaptarse muy bien al iniciar un proyecto y en equipos de trabajo pequeños de entre tres y nueve personas (Schwaber K., Sutherland J., 2020).

En una investigación realizada por Fahmideh M., Grundy J., Beydoun G., Zowghi D, Susilo W. and Mougouei D. (2022) se enfocaron en la RS para llevar el software legado a plataformas de la nube, como tal los métodos empleados sirven como base para una interoperabilidad efectiva y la adaptación de los sistemas en estas nuevas plataformas; el resultado obtenido fue un metamodelo personalizado para la reingeniería de plataformas en la nube denominado “Migration of Legacy Software Application to the Cloud” (MLSAC).

Con base a la revisión del estado del arte, se desea comprobar a través de una propuesta de modelo de RS que pueda favorecer a la mejora de la productividad y eficiencia en el mantenimiento de sistemas legados en el Tecnológico Nacional de México campus Zitácuaro.

## METODOLOGÍA

Para iniciar, el modelo propuesto se diseñó y aplicó a un sistema legado del Tecnológico Nacional de México campus Zitácuaro. Después de haber revisado la literatura sobre los modelos de reingeniería de software se propuso la combinación de algunas técnicas del modelo tradicional de RS, SCRUM y

CMMI con la finalidad de promover la mejora de procesos y aumentar la calidad por medio de la gestión del proyecto, el modelo se conforma por cuatro etapas: Evaluación, Abstracción, Ágil y Aplicación, véase Figura 3.

**Figura 3.** Modelo propuesto para la RS

**Para la etapa de evaluación se propone realizar cuatro actividades:**

1. Obtener los requerimientos funcionales del sistema legado mediante dos instrumentos, el primero un cuestionario de preguntas que se realiza al interior del equipo de trabajo y una bitácora para recabar datos generales del del sistema, véase Ilustración 1 y 2.
2. La segunda actividad consiste en revisar la documentación existente, tanto a nivel código, como manuales de programación, bases de datos etc., véase Ilustración 3.
3. La tercera actividad consiste en realizar una lista de cotejo para verificar que los módulos funcionen de acuerdo a lo esperado, véase Ilustración 4.
4. La cuarta actividad consiste en generar una lista con los módulos que van a ser actualizados y generar las observaciones correspondientes, véase Ilustración 5.

**Ilustración 1**. Cuestionario detección de requerimientos

**Formato 1. Cuestionario. Detección de requerimientos funcionales**

| Preguntas | Respuesta o comentario |
|---|---|
| ¿Cuál es el objetivo principal del software? | |
| ¿Qué problemas o necesidades específicas del usuario o el departamento espera que resuelva el software? | |
| ¿Quiénes serán los usuarios finales? | |
| ¿El sistema cuenta con un gestor de usuarios? Es decir, ¿maneja tipos de usuarios? ¿Es requerido? | |
| ¿Cuál es el alcance del proyecto? Es decir, ¿Hay algún límite en cuanto a lo que el software debe hacer o no hacer? | |
| ¿Qué plataformas se deben soportar? ¿Deben ser compatibles con varios sistemas operativos o dispositivos? | |
| ¿En cuánto al requerimiento, que debe validar el software? | |
| ¿Cómo se integrará el software con otros sistemas existentes? | |
| ¿Qué nivel de mantenimiento y soporte se requiere después del lanzamiento del software? Es decir, ¿qué tan constante puede llegar a recibir cambios? | |
| ¿cuál es el plazo para la entrega del software? | |
| ¿Tiene datos reales para realizar pruebas? | |
| ¿Se requiere algún reporte? | |
| ¿El sistema genera reportes o documentos? | |
| ¿Cuáles son las principales debilidades y problemas del software actual? | |
| ¿Cuál es el alcance de la reingeniería del software?, es decir, el desarrollo tendrá nuevos módulos a los existentes | |
| ¿Cuáles son los principales requerimientos funcionales que se deben abordar en la reingeniería del software? Es decir, ¿Qué módulos son los principales a analizar? | |
| ¿Se requerirá la migración de datos del software actual a la nueva versión? | |
| ¿Qué tecnologías y herramientas se utilizan en el sistema (en caso de existir código)? | |
| ¿Tiene relación con otro sistema? Y si es sí, ¿cuál es ese sistema y qué funcionalidad tiene? | |

**Ilustración 2.** Formato para la gestión del proyecto

**Formato 2. Formato para la gestión del proyecto**

**Datos del proyecto**

| | | | |
|---|---|---|---|
| **Nombre del Proyecto:** | a) | | |
| **Abreviatura** | b) | | |
| **Encargado del proyecto:** | c) | | |
| **Fase del Proyecto:** | d) | **Fecha de Creación:** | g) |
| **Documentado por:** | e) | | |
| **Revisado por:** | f) | **Fecha de Revisión:** | h) |

**Responsables**

| Nombre | Rol | Fecha | Correo Electrónico | Teléfono | Responsable de versión |
|---|---|---|---|---|---|
| i) | j) | k) | l) | m) | n) |
| | | | | | |
| | | | | | |
| | | | | | |

**Histórico**

| | |
|---|---|
| Nombre Requerimiento | o) |
| Objetivo | p) |
| Responsable | q) |
| Fecha de inicio | r) |
| Fecha fin | s) |

**Ilustración 3.** Formato para la evaluación de la documentación

**Formato 3. Evaluación de Documentación**

| Núm. | Pregunta | X | Observación |
|---|---|---|---|
| 1 | ¿Hay documentación disponible para el proyecto de software? | ☐ | |
| 2 | ¿La documentación está organizada y estructurada de manera clara?, es decir, separa código de documentación a nivel carpetas | ☐ | |
| 3 | ¿La documentación es fácil de leer y comprender?, maneja una estructura donde divide por secciones el tema a tratar | ☐ | |
| 4 | ¿La documentación cubre todos los aspectos importantes del proyecto, incluyendo la arquitectura, el diseño y la implementación? | ☐ | |
| 5 | ¿La documentación incluye ejemplos y casos de uso para ayudar a entender cómo funciona el proyecto?, estos se pueden encontrar en el manual de usuario | ☐ | |
| 6 | ¿La documentación está actualizada con los últimos cambios en el proyecto? | ☐ | |
| 7 | ¿La documentación incluye información sobre cómo instalar y configurar el software?, es decir, contiene algún manual de soporte | ☐ | |
| 8 | ¿La documentación describe los requisitos del sistema necesarios para utilizar el software? | ☐ | |
| 9 | ¿La documentación incluye información sobre cómo mantener y actualizar el software? | ☐ | |
| 10 | ¿El software tiene comunicación con otro sistema? Escribir en el espacio de observaciones con qué sistemas tiene comunicación | ☐ | |
| 11 | ¿El software maneja webservices o servicios rest? | ☐ | |
| 12 | ¿Se tiene el código de webservices o servicios rest? | ☐ | |

**Ilustración 4.** Formato de evaluación del sistema

**Formato 4. Evaluación del sistema**

| Preguntas | Respuesta | Observación |
|---|---|---|
| ¿El software cumple con todas las funciones descritas en la documentación? | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software se ejecuta sin errores?, es decir, muestra error en alguna función y qué tipo de error | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es fácil de instalar y configurar?, es decir, el proceso de instalación lo puede realizar quien sea | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es fácil de usar?, es decir, es intuitivo o fácilmente se encuentran las funciones prometidas | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es intuitivo y fácil de aprender? Es decir, se requiere alguna capacitación o con solo explicar su funcionamiento se puede usar | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es compatible con otros programas o sistemas operativos? | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software cumple con los requisitos del sistema descritos en la documentación? | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es seguro y protege los datos del usuario? Es decir, maneja algún tipo de encriptación de datos | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software funciona con rapidez y eficiencia?, funciona de manera ágil al consultar y realizar operaciones | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software es escalable y puede manejar grandes cantidades de datos o usuarios? | ☐Si<br>☐No<br>☐En algunos casos | |
| ¿El software proporciona suficiente retroalimentación al usuario?, es decir, maneja información que ayude al usuario a | ☐Si<br>☐No | |

**Ilustración 5**. Formato de lista preliminar

**Formato 5. Lista preliminar**

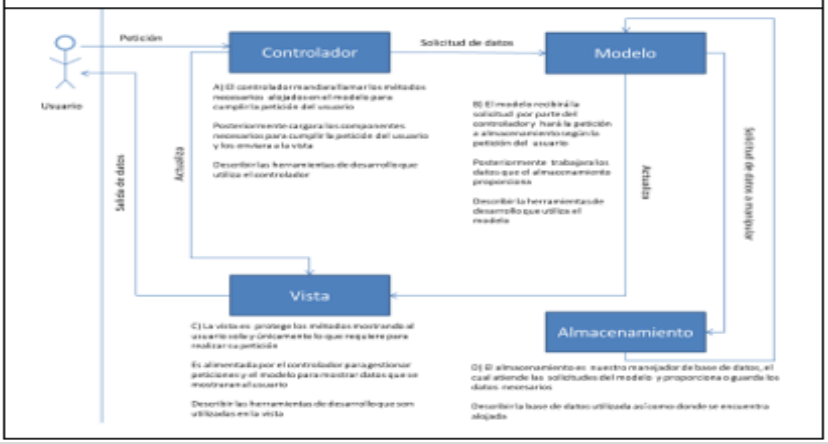
| Componente o módulo | Afectación (Si, No) | Observaciones |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

**Ilustración 6**. Ejemplo de diagrama de arquitectura.

**Ejemplo de llenado del Formato 6.**

**Diagrama de Arquitectura**

Instrucciones: Diseñar a mano alzada la arquitectura del sistema, considerando: usuarios, flujos, entidades: modelo, vista controlador, según sea el caso.



**La etapa de abstracción se compone por cinco actividades:**

1. Revisión de la arquitectura, en esta actividad se realizará un esquema que de forma gráfica presente el flujo de interacción entre usuarios, modelos, vistas y controladores, véase Ilustración 6
2. Análisis de la interfaz visual, en esta actividad se utiliza el método de observación, no será necesario aplicar algún instrumento, el responsable será responsable de verificar de acuerdo a los estándares actuales la satisfacción en cuanto a:
   a. La experiencia de navegación centrada en el usuario.
   b. La presentación o diseño de Interfaces.
3. Analizar los componentes y procedimientos, esta actividad utiliza el método de observación y documentación en código fuente del sistema para agilizar el mantenimiento de la programación.
4. Análisis de datos, consiste en entender el flujo de datos que existe en el almacenador de datos es parte importante del proceso para poder cumplir con el requerimiento solicitado, si en la documentación no existe un diagrama de datos.
5. Complementación, esta actividad se recomienda reunir al equipo para poder compartir los hallazgos encontrados en cada actividad.

**La etapa Ágil se conforma por tres actividades:**

1. Generación de Sprint, cada sprint puede tener al menos una semana de duración hasta tres semanas de acuerdo al requerimiento.
2. Plan de trabajo, en esta actividad se plasmará en un instrumento para llevar el control de las actividades a realizar.
3. Planificación de actividades, consiste en generar un calendario el cual debe incluir las fechas de entrega de cada sprint y las reuniones para dar avances del proyecto, estas pueden ser a consideración del equipo, lo recomendable es generar al menos una cada 15 días con una duración de 30 min, reuniones extraoficiales pueden llevarse a cabo para resolver dudas durante el desarrollo.

**La etapa de aplicación se conforma de tres actividades:**

1. Ejecución del Sprint,

   En esta etapa se aplica la ingeniería directa en la cual se realiza la codificación del sprint en curso, por lo cual es importante que el desarrollador agregue notas al código que elabora, al menos poner

una breve descripción antes de cada método. La codificación se debe estandarizar tanto en base de datos como en el código, es decir, determinar por medio de notación camello el nombrado de variables y métodos o procedimientos sin mencionar que deberán ser brevemente comentados o descritos antes de su declaración. Además de generar o reutilizar los procesos genéricos previamente analizados, teniendo como objetivo facilitar el mantenimiento y futuros desarrollos. Como propuesta de buenas prácticas, realizar el nombrado uniforme de carpetas que contengan código de la misma tipología, es decir:

*NombreSistema/JS/AngularJS/AdministradorUsuarios/*

*NombreSistema/JS/AngularJS/AltaDeUsuarios*

*NombreSistema/HTML/AdministradorDeUsuarios.html*

*NombreSistema/HTML/AltaDeUsuarios.html*

2. Team Testing, Al término del sprint antes de ser entregado se debe realizar una breve reunión al menos un día antes con el equipo de desarrollo con el fin de aprobar internamente en sprint y este no lleve errores.
3. Entrega de Sprint, en esta actividad basándose en el requerimiento o especificación funcional que se elaboró, dando como entrega el formato creado en la etapa ágil en la generación del sprint, validando que se cumpla lo establecido.

**RESULTADOS Y DISCUSIÓN**

Después de aplicar el modelo de RS propuesto para el caso de estudio, se determinó que es viable aplicarlo por los siguientes puntos:

- Las preguntas propuestas fueron de guía para poder extraer información al usuario. Anteriormente, estas entrevistas duraban de una a dos semanas, ya que no se tenía un objetivo sobre que preguntar. Al momento de aplicar la entrevista se redujo a solo dos sesiones, donde la primera se enfocó a la entrevista y la segunda para resolver dudas.
- Los formatos de evaluación ayudaron a entender las funciones básicas del sistema a pesar de no tener la documentación, así mismo también ayudaron a cuestionar el funcionamiento del proyecto, siendo esta la razón por la cual se extendió el desarrollo para tomar en cuenta las reglas de negocio de las demás áreas involucradas.

- Se generó y fomento la documentación para el proyecto, siendo un punto favorable para el desarrollo de futuros mantenimientos.
- La importación masiva datos fue una propuesta derivada del análisis correcto de requerimientos propuesto por el modelo en cuestión, el cual fue generado de manera que pueda ser reutilizado para otros módulos.
- Se obtuvo conocimiento de diferentes técnicas de desarrollo, gracias a la investigación realizada y expuesta al equipo.

El modelo propuesto enfatiza una interacción profunda con los usuarios mediante entrevistas estructuradas. Estas entrevistas, reducen drásticamente el tiempo que anteriormente podría haberse gastado en múltiples sesiones sin dirección clara. Esta técnica contrasta especialmente con el Modelo de Tradicional de RS, donde el inicio se basa en un estudio y documentación detallada del sistema existente, antes de embarcarse en una transformación y reconstrucción.

A diferencia del modelo propuesto, que pone un fuerte énfasis en la documentación esencial, Scrum, que es más un marco ágil que un modelo de reingeniería puro, se centra en un proceso iterativo e incremental. La estructuración dirigida de entrevistas y la creación de documentación no son centrales como en el modelo que se propone. El modelo propuesto, integra una interacción directa con los usuarios desde el principio, garantizando que las necesidades se comprendan y se aborden desde el inicio.

Aunque el modelo propuesto puede tener iteraciones implícitas, su énfasis no está en un ciclo repetido, sino en la claridad desde el comienzo mediante entrevistas y documentación.

Finalmente, una característica destacada del modelo es el enriquecimiento del equipo. A través de la investigación y el aprendizaje de los proyectos, el equipo no solo trabaja en el proyecto, sino que también mejora sus propias habilidades y conocimientos, algo que no se destaca tanto en los otros modelos mencionados. Basado en los resultados y contribuciones mencionados, es evidente que el modelo de reingeniería de software propuesto ha tenido un impacto positivo en el proceso de desarrollo y gestión de proyectos del Tecnológico Nacional de México campus Zitácuaro. Al Analizar de forma cualitativa los beneficios y contribuciones del modelo se tiene:

**Eficiencia en las Entrevistas**

Antes: Las entrevistas se alargaban por varios días debido a la falta de estructura y claridad en las preguntas.

Después: La estructuración de preguntas resultó en entrevistas reducidas a solo dos sesiones, lo que indica una comunicación más efectiva y eficiente.

**Documentación enfocada**

Antes: La falta de documentación era un problema para el desarrollo y mantenimiento del software.

Después: El modelo propuesto enfatizó la documentación, y se logró crear una que es concisa, enfocada y esencial para el mantenimiento futuro.

**Comprensión mejorada del Proyecto**

Antes: Había confusión o falta de claridad sobre ciertas funciones básicas del sistema.

Después: Los formatos de evaluación proporcionaron una herramienta esencial para entender y cuestionar el proyecto, incluso en ausencia de documentación.

Los puntos destacados anteriormente subrayan las áreas de mejora y optimización que el modelo introdujo en el proceso de desarrollo. Es evidente que el modelo puede actuar como una valiosa herramienta en proyectos similares, permitiendo una recopilación de requerimientos más estructurada, una documentación más robusta y una mejora general en la eficiencia del desarrollo de software.

La implementación del modelo no solo permitió la reingeniería del software en sí, sino que también facilitó el aprendizaje y la adquisición de diferentes técnicas de desarrollo por parte del equipo. Este conocimiento fue adquirido a través de la investigación y compartido con todo el equipo, potenciando sus habilidades y capacidades. La implementación exitosa del modelo propuesto en el Tecnológico Nacional de México campus Zitácuaro es un testimonio de su eficacia y de las contribuciones significativas que puede aportar al campo de la reingeniería de software.

## CONCLUSIONES

Con fundamento en la aplicación del modelo de reingeniería de software propuesto, se ha determinado que su viabilidad se sustenta en los siguientes aspectos positivos y beneficiosos:

Las entrevistas estructuradas permitieron una extracción eficiente de información.

Los formatos de evaluación resultaron útiles para comprender las funciones básicas del sistema, incluso en ausencia de documentación. Además, estos formatos brindaron la oportunidad de cuestionar el

funcionamiento del proyecto, lo que condujo a una extensión del desarrollo para tener en cuenta las reglas de negocio de las áreas involucradas. Esto ha mejorado la comprensión global del sistema y ha permitido una mayor coherencia en su implementación.

Se ha promovido y generado documentación adecuada para el proyecto, lo cual es beneficioso para futuros mantenimientos. El equipo de desarrollo ha logrado llenar los documentos esenciales de manera concisa y enfocada, sin perder de vista el objetivo del requerimiento.

En resumen, los resultados obtenidos a través de la aplicación del modelo de reingeniería de software demuestran su viabilidad. Los beneficios observados en la eficiencia de las entrevistas, la comprensión del sistema, la documentación, la reutilización de componentes y el crecimiento en el conocimiento del equipo de desarrollo respaldan la aplicación continua de este modelo.

**REFERENCIAS BIBLIOGRAFICAS**


Seacord, R. C., Plakosh, D., & Lewis, G. A. (2003). Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices 1st Edition. ISBN 0321118847

Medina A.S., Chaparro M. L (2013). Desarrollo de Aplicaciones WEB por Componentes – Código Libre. DOI: https://doi.org/10.22490/25394088.1092

Barbieri S. (2014). Framework de mejora de procesos de desarrollo de software. Recuperado de https://doi.org/10.35537/10915/4075 .

Bianchiotti F., Casas S. (2014). Guía para la reingeniería de sistemas legados: una experiencia práctica y real. DOI: https://doi.org/10.18294/relais.2014.99-106

Baldoneo J.A. (2017). Modelo CMMI y métodos ágiles en la gestión de proyectos software. Recuperado de http://hdl.handle.net/10651/43638

Aversano L., Grasso C., and Tortorella M. (2016). Gestionar la alineación entre los procesos de negocio y los sistemas de software. https://doi.org/10.1016/j.infsof.2015.12.009

Rodríguez M. V. 2017. Modelo CMMI y métodos agiles en la gestión de proyectos de software. Recuperado de: http://hdl.handle.net/10651/43638

I . Sommerville, Software Engineering, 9th edition, Pearson Publication, London, 2014.

M. Majthoub, M. H. Qutqut and Y. Odeh (2018). "Software Re-engineering: An Overview". 8th International Conference on Computer Science and Information Technology (CSIT), Amman,

2018, pp. 266-270, doi: 10.1109/CSIT.2018.8486173.

Gómez S.R. and Moraleda E. (2020). Aproximación a la ingeniería del software. Editorial: Editorial Centro de Estudios Ramon Areces SA. ISBN: 8499613292, 9788499613291

Bhavsar K., Shah V., Gopalan S. (2019). Machine Learning: A Software Process Reengineering in Software Development Organization. International Journal of Engineering and Advanced Technology (IJEAT). ISSN: 2249-8958, Volume-9 ISSUE-2, December, 2019.

Schwaber K., Sutherland J. (2020). The Scrum Guide. Recuperado de: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

Stojkov A., Kazi Lj., y Blazic M. (2020). Software Reengineering with Object-Oriented n-Tier Architecture: Case of Desktop-to-Web Transformation. International Conference on Information Technology and Development of Education – ITRO 2020 October, 2020. Zrenjanin, Republic of Serbia.

Singh J., Dhindsa K. S. and Singh J. (2020). "Performing Reengineering using Scrum Agile Framework,". *Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)*, Rajpura, India, 2020, pp. 33-35, doi: 10.1109/Indo-TaiwanICAN48429.2020.9181328.

Fahmideh M., Grundy J., Beydoun G., Zowghi D, Susilo W. and Mougouei D. (2022) A model-driven approach to reengineering processes in cloud computing. Volume 144, 2022, 106795, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2021.106795.