Daena: International Journal of Good Conscience. 13(2)239-258. Noviembre 2018.
ISSN 1870-557X

Ingeniería Inversa Caso de Estudio Sistema de Peaje

Treviño-Ibarra, Jonathan Baldemar, Lozano- Gonzalez Jorge & Torres –Guerrero Francisco

Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica, San Nicolás de los Garza, N.L., México. Email franciscot@gmail.com

Palabras Clave: Ingenieria de Software, Sistema de Peaje

Resumen

Investigación con respecto a la ingeniería inversa de sistemas Legacy con el manejo de base de datos de gran tamaño. El propósito de la investigación fue de investigar sobre proyectos en los cuales se utilizó la ingeniería inversa para así obtener un método certero para reproducirlo en un caso de prueba. Los sistemas Legacy se refieren a sistemas aún vigentes en la actualidad que utilizan tecnología obsoleta. En varios casos, los sistemas Legacy no cuentan con soporte para varios de sus componentes, o mucha de su documentación no existe o ha sido depreciada debido a los cambios realizados a los sistemas a lo largo de los años. Éste reporte de investigación presenta una indagación enfocada a la ingeniería inversa aplicada a dichos sistemas; se muestran varias metodologías que pueden ser aplicados a la ingeniería inversa de sistemas Legacy, y desarrolla uno en específico para la resolución de un sistema de peaje utilizado como ejemplo. Los procesos realizados a través de estas metodologías dieron como resultado nueva documentación de la arquitectura del sistema que detalla tanto su estructura como sus procesos.

Abstract

Research regarding the reverse engineering of Legacy systems with the management of large databases. The purpose of the research was to investigate projects in which reverse engineering was used to obtain an accurate method to reproduce it in a test case. Legacy systems refer to systems still in use today that use obsolete technology. In several cases, Legacy systems do not have support for several of its components, or much of its documentation does not exist or has been depreciated due to changes made to the systems over the years. This research report presents an investigation focused on reverse engineering applied to these systems; Several methodologies are shown that can be applied to the reverse engineering of Legacy systems, and develop one specifically for the resolution of a toll system used as an example. The processes carried out through these methodologies resulted in new documentation of the architecture of the system that details both its structure and its processes.

Introduccion

Los sistemas Legacy, también considerados como ‘heredados’, se refieren a sistemas realizados con tecnología que hoy en día resulta ser obsoleta y anticuada.

Los sistemas Legacy resultan ser un problema para el personal a cargo de su uso, debido a que dichos sistemas continúan siendo utilizados por las empresas, cuando hay alternativas más viables y convenientes para el uso del personal; las limitaciones impuestas por la utilización de sistemas heredados implican gran dificultad para aquellos que utilizan dicho sistema frecuentemente; asimismo, a medida que el tiempo y la tecnología avanzan, el

sistema heredado se va deteriorando cada vez más, hasta que llega un punto en el cual es necesario crear un nuevo sistema con tecnología más adecuada para usos modernos.

Un ejemplo puede ser aplicado en sistemas Legacy utilizados en departamentos informáticos gubernamentales; además de la utilización de sistemas heredados, comprometidos por el uso de tecnología obsoleta, el presupuesto utilizado para el mantenimiento de dichos sistemas está en declive, resultando en una mayor dificultad en el uso de dichos sistemas. Además del problema de presupuesto, cabe mencionar que es muy posible que mayoría, por no decir todos, los sistemas en los cuales depende el sistema Legacy no tienen soporte técnico hoy en día; un ejemplo es la utilización del sistema operativo Windows XP en dichos sistemas, el cual ya no tiene el beneficio de soporte técnico por parte de Microsoft hoy en día (TICbeat, 2015).

Marco Teórico

Un sistema Legacy se refiere a un método, tecnología o sistema de computación viejo o precedente a sistemas actuales. Un sistema Legacy es obsoleto en comparación con sistemas realizados con tecnología y/o métodos actuales.

La ingeniería inversa de bases de datos trata sobre los procesos de entender el funcionamiento de las bases de datos legacy y de extraer sus especificaciones de diseño (semánticas de dominio) (Chiang, Barron, & Storey, 1996). El proceso de ingeniería inversa de las bases de datos puede representarse como un proceso a partir del cual se obtienen las semánticas del dominio de una base de datos legacy; se realiza la transformación inversa del esquema de la base de datos (es decir, se pasa de un concepto lógico a uno conceptual) y, finalmente, se representa el resultado del proceso como un esquema conceptual.

Comparado con la ingeniería inversa estándar aplicada a un software, la ingeniería inversa sobre las bases de datos exhibe particularidades interesantes (Hainaut, 2009). En primer lugar, el propósito de la ingeniería inversa aplicada en bases de datos es el de recuperar la especificación completa de una base de datos en tal manera que su conversión a otro modelo de datos puede ser automatizado. En segundo lugar, utiliza una variedad de fuentes de información para su desarrollo, tales como DDL, análisis de código a análisis de datos, análisis de código de programa, observación del comportamiento del programa, y alineamiento de ontologías. Por último, se reconoce rápidamente que la ingeniería inversa de base de datos requiere técnicas de comprensión de programas, en la misma manera que la comprensión de programas que retienen una cantidad intensa de datos requieren de ingeniería inversa de base de datos.

Existen varias maneras de realizar la ingeniería inversa de bases de datos pertenecientes a un sistema Legacy. En el escrito *Digging Deep* (Strobl, Bernhart, & Grechenig, 2009) utilizaron ingeniería inversa sobre la base de datos de un sistema Legacy cuya documentación era obsoleta debido a los cambios realizados en el sistema a lo largo de los años. Dicho proceso se realizó en cuatro pasos:

- Diseñar un proceso de ingeniería inversa capaz de ser utilizado en un proceso de reingeniería de larga escala.
- Documentar el proceso en manera genérica y reutilizable
- Aplicar el proceso en el contexto del proceso de reingeniería
- Evaluar el proceso que el proceso tuvo en el proyecto, y los resultados obtenidos.

Para realizar éste proceso, en este caso de estudio se aplicaron 5 actividades, las cuales se ejecutaron de manera secuencial, para completar el proceso especificado.

1. *Análisis del estado actual*: Puesto a que no se conocía el tamaño ni la complejidad de la base de datos, se creó una vista general del sistema actual (en éste caso, fue asistido por herramientas CASE).
2. *Categorización de los Objetos dentro de la Base de Datos*: Se reconocen los objetos dentro de la base de datos que fueron cubiertos por el análisis previo. Los datos se clasificaron en tres categorías: *datos relevantes a negocios*, *datos específicos de sistema e implementación* y *aspectos obsoletos de la base de datos*. Se hace esto con el propósito de poder distinguir entre los aspectos que siguen siendo utilizados en el sistema de aquellos que ahora son obsoletos.
3. *Ingeniería Inversa de Datos*: Dicho proceso se comprende como el entendimiento semántico de los datos a ser migrados en el sistema rediseñado; es decir, comprender

el formato de los datos para reutilizarlo en el nuevo sistema. El documento indica que éste paso resulta ser el más intensivo en todo el proceso de ingeniería inversa.

4. *Análisis de Uso y Tráfico*: El último paso de ingeniería inversa. En éste paso se continúa monitoreando el funcionamiento del sistema para descartar falsos positivos y estimar goles de rendimiento del sistema.
5. *Presentación de Resultados*: Dicho paso se puede realizar después de cada uno de los pasos anteriores.

Otro método que puede ser utilizado es el de realiza ingeniería inversa a partir de diagramas UML existentes del sistema Legacy en cuestión, detallado en el documento realizado por Vukovic, Brdjanin y Maric (Vukovic, Brdjanin, & Maric, 2017). Para éste método, se asume que los elementos de la base de datos están definidos previamente y que están representados en notación UML.

Según el documento, el schema de la base de datos relacional está representado como el *paquete*, y todos los elementos que lo constituyen (*schemas relacionales*, *vistas* y *tipos de datos*) son reutilizados en tres subpaquetes (Tablas, Vistas y DataTypes). La manera de representar las relaciones entre las tablas es el uso de la dependencia que hace referencia al schema relacional del sistema.

Otro método de realizar este proyecto es mediante la re-ingeniería (J. Premerlani & R. Blaha, 1993), en el cual se explica un desarrollo de *ingeniería inversa* seguido por *ingeniería avanzada*. A través de ingeniería inversa, se toma un diseño o implementación pasada que representa un diseño y extrae de éste el contenido clave para lidiar con el problema en cuestión, además de desechar optimizaciones de diseño y decisiones de implementaciones.

Durante la ingeniería avanzada, el modelo de la esencia de la aplicación obtenido previamente se utiliza como la base para la implementación de un nuevo medio.

Con respecto a prácticas comúnmente realizadas en el proceso de ingeniería avanzada, se presentan algunas técnicas:

- Clases: Cada clase suele implementarse como una tabla, con una columna para cada atributo. Los índices únicos suelen especificarse en llaves primarias y llaves candidatas. Por último, una clase puede derivar su identidad de otra clase, u otras clases a través de una relación.
- Generalizaciones: El enfoque usual para las generalizaciones es de mapear cada clase dentro de una jerarquía a una tabla con columnas correspondientes a los atributos, y una llave primaria común. Una generalización genera particiones de datos; para cada registro en la tabla de superclase se encuentra un registro en una tabla subclase.
- Asociaciones: El constructo de asociación que más comúnmente se encuentra es el de la llave foránea sepultada, la cual se utiliza para implementar asociaciones binarias que no son de muchos a muchos. La llave foránea usualmente está enterrada en un fin de la asociación, sujeta a restricciones de duplicidad.

Con fines de conocer proyectos similares enfocados al desarrollo de un sistema de peaje, se investigaron varios proyectos enfocados a ese ámbito. Un proyecto investigado fue el del desarrollo de un sistema de peaje basado en un entorno de Linux en India (Desai & Patoliya, 2017). El proyecto fue realizado debido a que el sistema implementado anteriormente requería pago de peaje manualmente, lo cual ocasionaba graves congestionamientos en el tráfico. Se buscó desarrollar un sistema de peaje automatizado que evitara el congestionamiento causado por un sistema manual. El sistema con el cual sería reemplazado

sería un sistema basado en RFID, el cual está compuesto solamente de dos partes; un lector y una etiqueta. El sistema RFID automáticamente identifica y rastrea etiquetas que estén anexadas al objeto.

El lector RFID se comunica inalámbricamente con las etiquetas RFID para enviar y recibir información de las etiquetas. En el nuevo sistema diseñado, cada vez que un automóvil llegue a una cabina de peaje, el lector verifica la etiqueta RFID asignada al automóvil.

El sistema está planeado en ser diseñado e implementado en un entorno de Linux integrado a través de un Raspberry Pi el cual funciona como procesador principal, y una tarjeta SD es utilizada para contener el Sistema Operativo. El sistema se basa en detección de placas de licencias de automóviles utilizando procesamiento de imágenes. Una imagen de la placa de licencia de un automóvil, siendo un número de identificación único asignado para cada vehículo por las autoridades, es procesada por el sistema a través de una cámara web al pasar por la casilla de peaje. El número de licencia entonces es comparado con un listado de una base de datos proporcionado por el gobierno. El vehículo contendrá una cantidad nominal para pagar el peaje; una vez identificado, una cantidad de nómina será sustraída de la cuenta del usuario, y éste será notificado via mensaje SMS.

Método

Para fines de investigación, se obtuvo acceso a un sistema Legacy de peaje mediante la cooperación del cliente. Se obtuvo acceso puesto a que el sistema del cliente empezaba a mostrar problemas debido a la cantidad de datos que manejaba, ya que la tecnología obsoleta del sistema no era capaz de manejar la enorme cantidad de datos en circulación.

El sistema de peaje había sido desarrollado con tecnología que, en tiempos actuales, no es capaz de retener los datos que ha estado acumulando durante el tiempo en que éste fue implementado. Aunque la primera opción del cliente fue en ponerse en contacto con el personal involucrado en la realización de dicho sistema, esto resultó ser imposible debido a que la empresa encargada de la realización del proyecto se había disuelto. Al no tener personal que conozca el sistema, la opción que decidieron tomar fue la de realizar ingeniería inversa sobre el sistema de peaje. El cliente entonces nos contactó con propósitos de realizar documentación con respecto al funcionamiento del sistema en base a documentación ya existente pero obsoleta. Su propósito era poder realizar un nuevo sistema que funcione de manera igual al anterior y que maneje los datos de la misma manera, pero con el propósito de que ésta vez el cliente retuviera los planos y documentación de la arquitectura del nuevo sistema.

Para la implementación del nuevo sistema, se consideraron varias metodologías para utilizar en el proyecto. Dentro de las metodologías consideradas se encuentran las metodologías ágiles, las cuales adaptan la forma de trabajo dependiendo de las condiciones del proyecto a medida que éste se va desarrollando. Las metodologías ágiles buscan involucrar más al cliente en el desarrollo del proyecto, tratan de motivar más al equipo de desarrollo, busca eliminar costos y tiempo en el desarrollo y también mejora la eficiencia. Las metodologías

ágiles, en resumen, buscan alterar el proceso de desarrollo a medida que surgen cambios por necesidades.

Una de las metodologías ágiles investigadas es la metodología XP (Xtreme Programming). La metodología XP investigada puede ser aplicada a proyectos de mediana y grande escala; es decir, se puede aplicar a proyectos de aproximadamente 40 000 líneas de código en adelante (Rizwan & Qureshi, 2012). Se ha argumentado que la metodología XP normalmente no es aplicable a proyectos de éstas escalas debido a problemas con respecto a planificación estructural, su enfoque a resultados iniciales, documentación pobre y niveles bajos de cobertura de pruebas. El método XP le da más énfasis al lado de codificación/desarrollo y software operacional en vez de documentación comprensiva o diseño de arquitectura.

El diseño propuesto por Rizwan y Qureshi es una modificación al método XP existente para hacerlo aplicable a métodos de escala mediana y larga. Los pasos que sigue el modelo son:

1. Fase de Planificación del Proyecto: Los elementos que la componen son el juntar historias de usuario, bucles de retroalimentación, historias de estimaciones, criterios de aceptación de pruebas y planes de iteración. Sin embargo, los pasos anteriores no pueden ser aplicados a proyectos de escalas grandes. Por lo tanto, en vez de metas pequeñas, ésta etapa se debe de enfocar a grandes metas dentro del proyecto.
2. Fase Análisis y Gestión de Riesgos: En ésta fase, un analista reúne las historias de usuario y de requerimientos detalladas en la fase anterior. La fase de análisis produce historias de usuario y de requerimientos más comprensivas y una arquitectura sólida (es decir, una arquitectura flexible que permite patrones de diseño). El propósito de una estructura que permita utilizar patrones de diseño es cortar el costo y el tiempo que se puede utilizar en hacer nuevas historias de usuario o de requerimientos.

3. Fase Diseño y Desarrollo: Las fases de diseño y desarrollo del modelo XP existente se unen en éste nuevo diseño para hacer un proceso más adaptativo. El proceso de XP propuesto utiliza ciclos de prototipos y demos para verificar el diseño y las historias de usuario. El software es desarrollado en pequeñas versiones o incrementos a medida que el cliente aprueba de los prototipos. La tarea de codificación es asignada a programadores bajo la programación en pares.
4. Fase de Pruebas: Los casos de pruebas son preparados antes de la codificación de una nueva versión, siguiendo un entorno de desarrollo impulsado por pruebas (TDD). Cada versión es evaluada basada en la unidad individual. La prueba de integración se realiza para probar la integración entre los módulos individuales. Las pruebas del sistema es un paso para validar todos los incrementos como una sola unidad. Las pruebas de aceptación son el último paso para verificar un lanzamiento destinado para el cliente. Un lanzamiento probado es mantenido e implementado. El objetivo principal de una estrategia cíclica es mejorar y cambiar las actividades requeridas en el desarrollo mediante la reorganización de acciones adicionales requeridas como resultado de las pruebas.

Otra metodología tipo ágil dedicada al desarrollo de software es Scrum. La metodología Scrum está destinada a ser utilizada en proyectos de estructura compleja; Scrum aborda proyectos de manera que éstos se desarrollan de manera dinámica y flexible, pretendiendo que se mantenga una continua interacción con el cliente. (Martínez, 2013)

La metodología Scrum recae en cuatro roles principales para su desempeño:

- *Product Owner*: Representa a la voz del cliente y aquellos interesados en el proyecto, pero no implicados en su desarrollo. Define los objetivos del proyecto y garantiza el cumplimiento de dichos objetivos.
- *Scrum Master*: Encargado del Scrum Team; garantiza que el equipo de trabajo no tenga dificultades durante el desarrollo del proyecto.
- *Scrum Team*: Equipo encargado del desarrollo del proyecto.
- *Stakeholders*: Perfiles de aquellos interesados en el proyecto, tales como directores, dueños, comerciales, etc. ajenos al cliente y al equipo de desarrollo.

Ya teniendo identificado a los componentes del equipo de trabajo de Scrum, se puede denotar que el proceso de desarrollo de scrum se compone de los siguientes pasos:

1. Elaboración del Product Backlog: El Product Backlog se trata de un archivo el cual contiene las tareas, requerimientos y funcionalidades que el proyecto necesita. La persona que tiene autoridad sobre cambios correspondientes al Product Backlog es el Product Owner.
2. Definicion del Sprint Backlog: Al tener listas las tareas a desempeñar en el proyect, el Sprint Backlog trata de repartir las tareas a personas en el Scrum Team. Además de asignar las tareas, se le asigna a cada tarea una estimación de tiempo a través de un coste. Si el coste es muy grande, las tareas se dividen en tareas cada vez más pequeñas para poder realizarlas.
3. Elaboración de Sprint: Se trata del periodo de desarrollo de las tareas previamente repartidas. Se realizan entregas parciales para ir realizando pruebas. Éste paso se continúa realizando hasta que todos los elementos del Backlog han sido entregados.

4. Estado de Burn Down Chart: Durante la elaboración del Sprint, el Burn Down Chart se marca la evolución de las tareas y los requerimientos que aún no han sido realizados.
5. Sprint Planning Meeting: Se realizan reuniones de trabajo en el cual el Product Owner planifica procesos para cumplir con las tareas contenidas en el Product Backlog. Dentro de éstas reuniones también se encuentra el Daily Scrum, los cuales (como su nombre lo indica) son juntas diarias que delinean los objetivos del día siguiente y se analizan los problemas que se han encontrado hasta ahora, y maneras de cómo resolverlos con el resto del equipo.

Experimentación

El método utilizado durante el desarrollo de la ingeniería inversa fue un método similar al método indicado en el escrito *Digging Deep* (Strobl, Bernhart, & Grechenig, 2009). Sin embargo, a diferencia del método presentado en el texto anterior, el método que se utilizó para la ingeniería inversa del sistema de peaje sólo fue utilizado hasta el tercer paso debido a que el cliente sólo nos contrató para realizar documentación con respecto a la arquitectura del sistema y prototipos iniciales, y no para el desarrollo completo de una nueva aplicación.

1. *Análisis del estado actual*

En los estados iniciales del proyecto, se realizaron varias visitas con el propósito de conocer el estado actual del software; se obtuvo la documentación más actualizada de todo el sistema,

se verificaron las tablas que se encuentran en la base de datos en la actualidad y la estructura de la base de datos.

Durante un periodo de tiempo se verificó la información obtenida de las visitas; se revisó la documentación obtenida y se descartó la documentación con información obsoleta. Asimismo, con la información obtenida de las tablas dentro de la base de datos, se obtuvo un listado completo de las tablas existentes en la base de datos, así como la estructura de dichas tablas.

Al ya haber obtenido la totalidad de las tablas dentro de la base de datos, el siguiente paso a seguir fue separar las tablas que tienen contenido dentro de la base de datos de las tablas sin utilizar.

Se realizaron documentos actualizados tomando en cuenta la información obtenida; algunos ejemplos de documentos realizados fueron Diagramas de Entidad – Relación actualizados, Diccionario de Datos e Información General de la base de datos

2. *Categorización de los Objetos dentro de la Base de Datos*

Testeo de mensajes de dispositivos; descubrimiento de las tablas implicadas en los procesos de los dispositivos

Aunque se obtuvo una documentación actualizada de la base de datos, el propósito de las tablas seguía sin ser comprendido; la información proporcionada por la documentación anterior sólo abarcaba la estructura de las tablas, y muy pocas tablas mencionaron el propósito en el cual se utilizarían dentro del funcionamiento del sistema. Por lo tanto, lo que

se realizó fue investigar el funcionamiento de los procesos del sistema en tiempo real, y obtener las tablas involucradas en los procesos a partir de ésta observación.

Los procesos realizados en el sistema provienen principalmente de 4 dispositivos; para propósitos de explicación del sistema, se les denotará como Expendedora, Torniquete, Inicializadora y Personalizadora. Cada dispositivo maneja procesos al realizar sus funciones, y cada dispositivo deja mensajes de comunicación entre los dispositivos y la base de datos. Los mensajes que manejan los dispositivos varían con varias razones, tales como Alarmas, Advertencias, Ventas, Eventos, entre varios otros; aunque varios mensajes son universales en todos los dispositivos (como Alarmas y Advertencias), algunos otros sólo son vistos en los dispositivos especializados a los que pertenecen.

Ya explicado esto, lo que se realizó durante éste proceso fue el inducir los procesos que realiza cada maquinaria y, durante el proceso de inducir los procesos, vigilar los mensajes que envían los dispositivos al sistema. Los mensajes que cada dispositivo envía son un indicativo del tipo de proceso que el dispositivo realiza al hacer una acción; asimismo, los mensajes contienen una fecha de expedición desde los dispositivos. Utilizando algunos datos identificables de los mensajes, se verificaron tablas basándose en la fecha de inserción y varios datos que deben de ser similares a los datos expedidos por el mensaje. Dentro de la base de datos se identificaron un cierto número de tablas que fueron modificadas con los datos que utilizamos como indicadores, por lo que se identificaron las tablas involucradas en los procesos y su funcionamiento en dichos procesos.

Ya obtenidas las tablas involucradas en los procesos, lo que se realizó a continuación fue realizar un Modelo del Dominio de las tablas en el sistema; lo que se realizó fue separar las tabas en entidades que realizan varias funciones diferentes en el sistema.

3. *Ingeniería Inversa de Datos*

Después de haber separado las tablas involucradas en procesos del sistema, nuestra atención se enfocó en otro detalle: aunque se conocen ciertos datos de los mensajes, no se conocen bien los datos adicionales contenidos en los mensajes. Después de identificar las tablas relacionadas a los tipos de mensaje, lo que siguió fue identificar la arquitectura de los mensajes.

Cada mensaje, dependiendo de su tipo, varía en respecto a su arquitectura y la cantidad de información que poseen; por ejemplo, un mensaje de Alarma es de longitud corta, mientras que un mensaje de Ventas es de longitud grande pues contiene una gran cantidad de información. Dicho esto, cada mensaje realizó un registro dentro de la base de datos en varias tablas, lo cual se mencionó fue identificado en el último paso; lo que se realizó a continuación fue comparar los registros creados por los mensajes con los mensajes mismos para identificar las porciones de los mensajes que escribían sobre los campos en las tablas.

Se realizó éste proceso con varios mensajes de prueba para poder verificar la tendencia en la arquitectura entre todos los mensajes y obtener una arquitectura común de cada tipo de mensaje; además, se aislaron casos extraños en los cuales la arquitectura del mensaje variaba y se identificó la razón de porqué los mensajes diferían con la norma encontrada. El proceso de parseo de cada uno de los mensajes y la formación de la arquitectura de cada uno se documentó en un archivo de Flujo de Información, el cual fue creado para cada uno de los dispositivos.

Resultados

Como resultado de la investigación que se detalló a través de éste documento, se realizaron un conjunto de documentos que ilustran la arquitectura del sistema Legacy de peaje manejado por el cliente. Los documentos redactados como resultado de la investigación del sistema fueron: el **Diagrama Entidad-Relación** completo de la base de datos, el cual detalla las tablas dentro de la base de datos; el **Diccionario de Datos**, el cual detalla la estructura de las tablas existentes dentro de la base de datos, y en el caso de las tablas que aún siguen en uso en el sistema actual, un ejemplo de su arquitectura; **Flujos de Información** pertenecientes a cada dispositivo manejado por el sistema, los cuales detallan los mensajes enviados durante los procesos de los dispositivos y cómo se interpretan con respecto a la base de datos; un reporte con respecto a la función de **Listas Negras**, la cual detalla los procesos que se realiza al insertar una tarjeta dentro de la base de datos, así como las *Telecargas* que son necesarias realizar para actualizar las listas en los dispositivos; y por último, un documento de Modelo de Dominio el cual demuestra la relación entre las tablas de la base de datos dentro del sistema – las tablas se dividen en entidades las cuales interactúan con otras entidades de acuerdo a los procesos que se lleven a cabo en el sistema.

Conclusiones

En éste reporte de investigación se demostró los que son los sistemas Legacy, la importancia que tiene mejorarlos y la necesidad de realizar ingeniería inversa sobre aquellos sistemas con los cuales ya no se cuenta con soporte.

Para el ejemplo propuesto de un sistema Legacy de un sistema de peaje, se investigaron varios procesos realizados por otros ingenieros para realizar ingeniería inversa a sistemas Legacy y a bases de datos utilizadas en dichos sistemas. Asimismo, se investigó un caso de desarrollo enfocado a un sistema de peaje para basarse en ese caso para la realización de ingeniería inversa del proyecto actual.

Como resultado de la investigación de métodos de ingeniería inversa, se obtuvo un método apropiado para proceder con el sistema actual, puesto a que se pudo reproducir junto con la documentación que actualmente existía del sistema. Se obtuvo la información válida del documento, se verificaron las partes que aún seguían siendo utilizadas en el sistema actual, se realizaron varias pruebas para determinar las funcionalidades del sistema y cómo éstas modificaban a la base de datos, y se realizó una investigación de cómo funcionaron dichos procesos.

Como resultado de la investigación realizada en el sistema Legacy, se obtuvo una documentación actualizada con respecto a las funcionalidades del sistema, un panorama actualizado de la base de datos, y una vista actualizada de los procesos y las tablas que se afectan en la base de datos.

Bibliografía

1. Chiang, R. H., Barron, T. M., & Storey, V. C. (1996). A framework for the design and evaluation of reverse engineering methods for relational databases. *Data & Knowledge Engineering 21*, (págs. 57-77).

2. Desai, M. M., & Patoliya, J. J. (2017). Smart toll collection system using embedded Linux environment. *2nd International Conference for Convergence in Technology (I2CT)* (págs. 79-83). Mumbai, India: IEEE.

3. Hainaut, J.-L. (2009). Legacy and Future of Data Reverse Engineering. *16th Working Conference on Reverse Engineering* (pág. 4). Lille, Francia: IEEE.

4. J. Premerlani, W., & R. Blaha, M. (1993). An Approach for Reverse Engineering of Relational Databases. *GE Corporate Research and Development* (págs. 151-160). Baltimore, MD: IEEE.

5. Martínez, E. (30 de Mayo de 2013). *iebschool.* Obtenido de https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

6. Rizwan, M., & Qureshi, J. (2012). Agile software development methodology for medium and large projects. (págs. 358–363). Arabia Saudita: IET Software.

7. Strobl, S., Bernhart, M., & Grechenig, T. (2009). Digging Deep: Software Reengineering Supported by Database Reverse. *IEEE International Conference on Software Maintenance* (págs. 407-410). Edmonton, Canada: IEEE.

8. TICbeat, R. (6 de Abril de 2015). *TicBeat.* Obtenido de http://www.ticbeat.com/tecnologias/sistemas-heredados-obstaculo-para-gobiernos-de-todo-el-mundo/

9. Vukovic, D., Brdjanin, D., & Maric, S. (2017). A UML-based approach to reverse engineering of relational databases. *25th Telecommunications forum TELFOR 2017* (págs. 1-4). Belgrade, Serbia: IEEE.