Reingeniería de software aplicada a sistemas heredados

Felipe de Jesús Vidaña Mireles

Instituto Tecnológico de la Laguna

felipe_mv1545@hotmail.com

Resumen

Existen diferentes metodologías para aplicar el proceso de la Reingeniería de Software a sistemas informáticos que por alguna causa requieren de una actualización o de un mantenimiento perfectivo. Estas aplicaciones deben ser restructuradas o reconstruidas bajo un proceso de Reingeniería de Software. Está a portación propone una metodología de Reingeniería de Software para sistemas heredados. Y proporciona una directriz para su aplicación. La metodología de reingeniería de software se define a partir de los problemas detectados en los sistemas informáticos. Está compuesta de procesos en los que involucra actividades específicas y dependiendo de la magnitud de los sistemas informáticos esto requerirá el tiempo y esfuerzo empelado. Cuando se detecta y analiza la problemática que afecta a los sistemas informáticos, se identifica y se aplica la para ese sistema Y así garantizar la correcta actualización del sistema que ahora deberá de ser más flexible y mantenible.

Palabras Clave: Sistemas heredados, mantenimiento perfectivo, reingeniería de software.

Abstract

There are different methodologies to implement the process reengineering software to computer systems for any reason require an update or a perfective maintenance. These applications must be restructured or rebuilt under a Software Process Reengineering. This article proposes a methodology of software reengineering legacy systems. The

software reengineering methodology is defined based on the problems identified in computer systems. It consists of processes involving specific activities and depending on the magnitude of computer systems this will require time and effort. When it detects and analyzes the problems affecting computer systems, identifies and applies for that system and thus the new system will be more flexible and maintainable.

Keywords: *legacy systems, perfective maintenance, software reengineering.*

1. Introducción

Hoy en día los sistemas informáticos se han vuelto muy importantes en nuestra vida cotidiana, prácticamente los usamos todo el tiempo, desde hacer cosas básicas tales como comunicarnos, informarnos, ver películas, ver televisión, realizar investigaciones o consultas, etc. Pero sobretodo ha tenido un gran impacto en las industrias en donde la mayoría de las empresas utilizan algún sistema informático para procesar información, realizar procesos, tareas etc. Conforme transcurre el tiempo los sistemas informáticos van evolucionando y aquellos sistemas informáticos que se desarrollaron con una antigüedad aproximadamente de 10 años o más, son conocidos como sistemas heredados. En la actualidad hay demasiados sistemas heredados, en donde los lenguajes de programación con que fueron desarrollados van quedando obsoletos, y estos sistemas heredados necesitan ser reconstruidos o reestructurados en un nuevo lenguaje de programación o que necesiten ser migrados hacia otro entorno para que continúen operando. Entonces estas actividades forman parte de la manutención de los sistemas heredados, lo cual es fundamental aplicarlo para mantenerlos en funcionamiento. Entonces la reingeniería de software surge de esta necesidad, en donde las empresas han invertido mucho dinero en sistemas informáticos, y para obtener algún beneficio de coste buscan que los sistemas informáticos operen por años.

2. Métodos

La reingeniería de software examina los sistemas heredados con el objetivo de reestructurarlos o reconstruirlos de manera que demuestren una mayor calidad. La reingeniería de software está enfocada hacia las actividades de mantenimiento, en el cual los objetivos son entendimiento, reparación, mejoramiento y evolución. Se describe una metodología de reingeniería de software en donde se aplica a los sistemas heredados que los cuales necesiten ser reestructurados o reconstruirlos y migrarlos hacia un nuevo entorno, es decir, los sistemas heredados son comúnmente aplicaciones de escritorio y por una u otra razón van quedando obsoletos y necesitan un mantenimiento evolutivo, es decir para mejorar y añadir nuevas funcionalidades, entonces esta sería una situación donde se podría aplicar la reingeniería de software.

Hay algunas pre-condiciones para aplicar la reingeniería de software, por ejemplo cuando no hay documentación, los sistemas están sujetos a cambios frecuentes que pueden afectar parte del diseño, y sobre todo cuando los sistemas cumplen con los requisitos y rendimiento requerido, esta precondición es debido a que no tendría caso reestructurar un sistema de pobre calidad y que no realice las operaciones solicitadas, en todo caso sería recomendable desarrollarlo nuevamente desde cero. Mencionadas la pre-condiciones anteriores a continuación se menciona una metodología ya que Debido a la inexistencia de una definida para la reingeniería de software y en base a investigaciones realizadas sobre este tema, se propone una metodología que ayude al proceso de restructuración y reconstrucción de aplicaciones heredadas hacia entorno web, que se muestra en la figura 1.

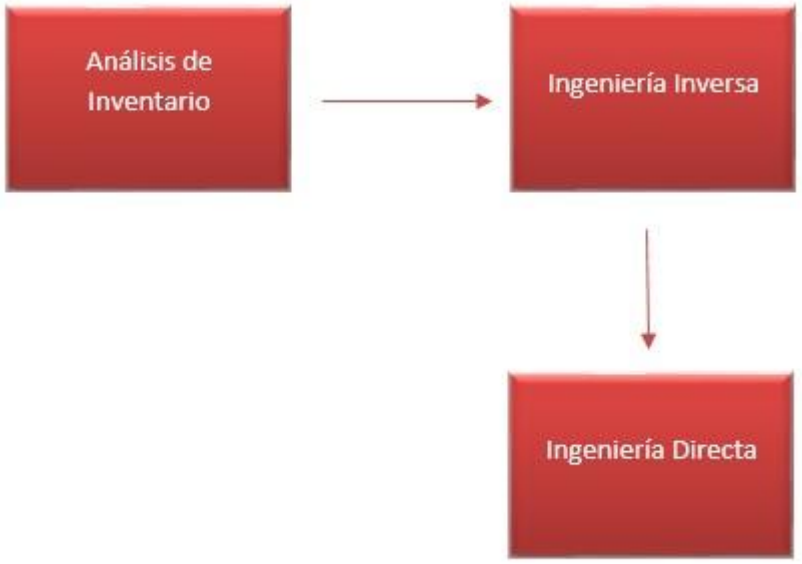

Figura 1 Metodología de Reingeniería de Software.

Antes de iniciar los procesos que se muestran en la figura 1, primero se tiene que analizar y comprender los sistemas heredados, así como los procesos, actividades y tareas que se realizan en la empresa, de los cuales tengan relación con los sistemas heredados.

El primer pasó realizar un inventario de los componentes de todos los formularios de los sistemas heredados, este proceso recaba información que proporciona una descripción detallada, por ejemplo: tamaño, antigüedad, importancia para el negocio, etc., de todas las aplicaciones activas. Se elabora un documento con los siguientes apartados: Denominación, Funcionalidad, Desarrolladores del Sistema, Fecha de Desarrollo, Descripción del Sistema, enumeración de ficheros (fuentes y objeto), tablas y de datos.

El segundo paso es aplicar ingeniería inversa a los sistemas heredados, con el propósito de crear una representación de los sistemas con un nivel de abstracción más elevado que el código fuente, de manera que se analiza el programa y se extrae información de él, ya que ayuda a documentar su organización y funcionalidad. Este es

el proceso de analizar el software con el objetivo de recuperar su diseño y especificación. Para llevar a cabo este proceso se utilizará un enfoque basado en 5 vistas, cada vista servirá para recuperar cierta información de diseño. Las cuales se muestran a continuación:

1.- Identificación de las tareas funcionales del sistema.

2.- Recuperación de la vista de diseño

3.- Representación de la interacción entre los objetos

4.- Identificación de componentes de software

5.- Identificación de componentes para el despliegue

El uso de la ingeniería inversa para reconstruir los aspectos arquitectónicos de los sistemas heredados puede referenciarse como redocumentación estructural o recuperación arquitectónica. Como resultado, se deriva la visión global de los sistemas heredados y se pueden recapturar algunos aspectos de su diseño arquitectónico.

Figura 2 Descripción del modelo desde 5 vistas.

Con esto se presenta un enfoque de arquitectura de sistemas basado en vistas. Estas vistas muestran los diferentes aspectos del sistema que son modelados. Cada una de estas vistas servirá para realizar una tarea específica, para la descripción del contenido de cada una de estas vistas usaremos los diagramas. El estándar que hemos adoptado es el UML (Lenguaje de Modelamiento Unificado) ya que proporciona un amplio conjunto de diagramas que normalmente se usan en pequeños subconjuntos para poder representar las cinco vistas principales de la arquitectura de un sistema. En la siguiente tabla se muestra las actividades a realizar, junto con las vistas y diagramas de UML que corresponden a cada actividad.

Tabla 1 Vista general de las actividades en la etapa de reingeniería inversa.

Actividad	Vista	Diagrama
Identificación de requerimientos funcionales	Casos de Uso	Casos de Uso
Recuperación del diagrama de clases	Diseño	Clases
Representación de la interacción entre objetos	Procesos	Secuencia y Colaboración
Identificación de los componentes de software	Implementación	Componentes
Identificación de los componentes para el despliegue	Despliegue	Despliegue

Actividad 1: Identificación de requerimientos funcionales. Para realizar esta actividad se hace uso de los diagramas de casos de uso. En la ingeniería directa, el análisis se enfoca en la funcionalidad y luego el diseño agrega los detalles específicos de la implementación y los aspectos relacionados con los requerimientos no funcionales. En la ingeniería inversa, todos los detalles de diseño e implementaron ya están definidos, de esta manera el límite entre el análisis y el diseño es bastante difuso. En la ingeniería inversa, los casos de uso permiten descubrir, en un alto nivel, que hace el sistema desde el punto de vista del usuario.

Actividad 2: Recuperación del diagrama de clases. La vista de diseño soporta principalmente los requisitos funcionales del sistema, es decir, los servicios que el sistema debe proporcionar. Es en esta vista donde se obtiene una visión global del funcionamiento del sistema. Se establecen los primeros requisitos funcionales en base a lo que se espera conseguir. En esta etapa se identifican, dentro del código, las clases que realizan o soportan las acciones descritas en los casos de uso.

Actividad 3: Representación de la interacción entre los objetos. La vista de procesos se utilizara para especificar operaciones que son ejecutadas por cada una de las clases identificadas en la vista de diseño y el flujo de colaboración entre ellas. Los casos de uso identificados sirven de guía para encontrar, dentro del código fuente, los objetos que intervienen en el proceso.

Actividad 4: Identificación de componentes de software. La Vista de implementación describe la organización estática de los módulos de software (código fuente, componentes, ejecutables) en el ambiente de desarrollo en términos de paquetes, de capas y de la administración de la configuración.

Esta vista puntualiza una correspondencia entre los elementos de la vista de diseño y las entidades físicas que encapsulan su implementación. También se indican las características físicas de los componentes, tales como: memoria requerida, nombre con el que se registra el componente en el sistema, versión y modelo del componente,

tiempo de servicio de cada operación, etc., necesarios para realizar la evaluación de la arquitectura.

Actividad 5: identificación de componentes para el despliegue. La vista de despliegue de un sistema contiene los nodos que forman la topología hardware sobre la que se ejecuta el sistema. Se preocupa principalmente de la distribución, entrega e instalación de las partes que constituyen el sistema. Los aspectos estáticos de esta vista se representan mediante los diagramas de despliegue y los aspectos dinámicos con diagramas de interacción, estados y actividades.

Una vez teniendo el diseño del sistema recuperado a partir de los pasos anteriores, el siguiente paso es aplicar ingeniería directa que incluye todas las actividades que se realizan para transformar el software existente en un software diferente, y más fácil de mantener, entre ellas están: descomposición, reestructuración, remodularización, redocumentación manteniendo su misma operatividad pero ahora con mayor usabilidad, mantenibilidad y funcionalidad.

Estas metodologías son aplicables cuando los sistemas heredados van a ser restructurados a un entorno web, Cuando ya se tiene la metodología definida y el análisis previo a los sistemas heredados, el siguiente paso es definir las herramientas de desarrollo, a continuación se describe cada una de ellas.

Php: se recomienda usar este lenguaje de programación porque es gratuito y no tendría gran impacto en el coste de la metodología. Este es un lenguaje orientado a objetos, en el cual se podría aplicar patrones de diseño y de esta manera el nuevo sistema sería más mantenible y adaptativo a los posibles requerimientos funcionales en el futuro.

Css: es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas. Es posible usar HTML, o incluso abusar del mismo, para añadir formato a los sitios web. Sin embargo, CSS ofrece más opciones y es más preciso y sofisticado. CSS está

soportado por todos los navegadores hoy día. Por lo tanto dar estilo y formato a las nuevas interfaces ofrece usabilidad a los usuarios finales.

JavaScript: es un lenguaje se utilizaría para conectar la programación del lado del cliente con la programación del lado del servidor.

JQuery validate: es un plugin que sirve para hacer la validación de formularios fácil y simple, ofrece varias opciones de personalización. Las validaciones de los formularios es un punto fundamental en cuanto al tema de seguridad, porque con el uso de este plugin se cuida la filtración de datos erróneos, o datos que no correspondan al campo requerido. El plugin viene con un útil conjunto de métodos de validación, incluida la URL y validación de correo electrónico, mientras que proporciona una API para escribir métodos personalizados.

Ajax: Técnica de desarrollo web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes.

En aplicaciones AJAX se pueden enviar peticiones al servidor web para obtener únicamente la información necesaria, empleando un lenguaje para servicios web basado en XML, y usando JavaScript en el cliente para procesar la respuesta del servidor web. Esto redunda en una mayor interacción gracias a la reducción de información intercambiada entre servidor y cliente y a que parte del proceso de la información lo hace el propio cliente, liberando al servidor de ese trabajo. La contrapartida es que la descarga inicial de la página es más lenta al tenerse que bajar todo el código JavaScript.

3. Resultados

Aplicar la metodología de reingeniería de software mencionada, se presenta la oportunidad de innovar en sistemas heredados que con el paso de los años van teniendo problemas de obsolescencia, solo por mencionar algunos, y esto es debido a que los lenguajes de programación no son estáticos y van cambiando con el paso de

los años. Si nos remontamos años atrás solamente había aplicaciones de escritorio, eran muy escasas las ideas de desarrollar aplicaciones web, sin embrago en la actualidad han tenido mucha demanda, sobre todo por la facilidad de acceso, ya que ahora no solo se accede desde la computadora, si desde diferentes dispositivos tales como móviles, ipad, etc. Entonces aplicando las metodologías mencionadas se pretende dar manutención a los sistemas heredados, innovándolos, manteniendo su calidad y cumpliendo con los requisitos de los usuarios. El tema de discusión que ha surgido sobre este tema, es que para algunos ingenieros de software no es obligatorio utilizar reingeniería software y prefieren desechar los sistemas heredados y empezar a desarrollarlo desde cero, obviamente esto tendría un impacto en el tiempo y coste, y pues prácticamente las empresas en general no invierten mucho dinero en el área de informática. Entonces para los ingenieros de software que optan por implementar la metodología de reingeniería de software mencionada en este artículo, tendrán mucho más posibilidades de encontrar proyectos donde se necesite actualizar los sistemas informáticos.

4. Bibliografía

Bianchiotti, F. (2014). Guía para la Reingeniería de Sistemas Legados. Rio Gallegos.

Juan Carlos Álvarez García, M. M. (2004). METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACION DE APLICACIONES CIENTIFICAS HEREDADAS. Salamanca.

Ricse, J. J. (2007). Ingeniería inversa aplicado a sistemas desarrollados para obtener documentacion. Lima.

Sommerville, I. (2001). Software Engineering. Sixth edition. Addison Wesley.

Pressman, R. S. (2002). Ingenieria del software. Mc. Graw Hill.