Guía para la Reingeniería de Sistemas Legados: Una Experiencia Práctica y Real

Franco Bianchiotti
Departamento de Informática
Administración General de Vialidad Provincial
Río Gallegos, Argentina
francofjb@gmail.com

Sandra Casas
Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Río Gallegos, argentina
scasas@unpa.edu.ar

***Resumen*—Este trabajo presenta una experiencia real de reingeniería de sistemas legados. La misma tuvo lugar en la Administración General de Vialidad Provincial (Santa Cruz) y demandó aproximadamente un año de trabajo. El proyecto de reingeniería tuvo dos objetivos muy concretos: a) desarrollar un marco de trabajo de reingeniería que se adecuara a los sistemas legados de la organización con la finalidad de poder ser replicado a otros sistemas legados de la organización. Este objetivo dio por resultado la elaboración de una guía para su posterior reutilización a estos contextos. b) su aplicación a uno de los sistemas legados de la organización. En esta primera instancia se aplicó al sistema de Patrimonio.**

Palabras Clave—Sistemas legados, reingeniería, migración, mantenimiento.

I. INTRODUCCION

Pragmáticamente los sistemas legados se han definido como "software que es vital para la organización, pero no se sabe qué hacer con él" [1]. Los sistemas legados son uno de los principales problemas del mantenimiento de software. A pesar que la crisis Y2K forzó a muchas organizaciones a sustituir los sistemas antiguos, existe aún software que supera los 20 años. Se trata de software monolítico, dividido en subsistemas que cumplen funciones importantes para las organizaciones, pero de los cuales no se dispone documentación, los desarrolladores originales ya no trabajan en la organización y solo se dispone de los códigos fuentes e incluso en algunos casos ni siquiera eso.

Aunque tecnologías más modernas (aplicaciones cliente-servidor, BD relacionales, tecnologías OO / componentes, etc.) suponen mayor facilidad de mantenimiento y además ofrecen mas oportunidades, ocurre que el ciclo de vida de los sistemas se extiende más de lo debido. Sneed [2] afirma que las empresas no invierten recursos en ingeniería inversa o reingeniería a las nuevas tecnologías sobre la base de que los costes de mantenimiento son menores. Sin embargo, en un punto el mantenimiento adaptativo [3] obliga a realizar el mayor esfuerzo y más riesgoso, el desarrollo de un nuevo sistema, la migración de los datos, la recuperación de la lógica de negocios, es decir una reingeniería [4]. Los procesos de reingeniería son complejos, y no existen formulas preestablecidas que se puedan aplicar con éxito a todos los casos, por otro lado conllevan una importante carga de trabajo manual, aún cuando se puedan utilizar herramientas que automaticen ciertas actividades.

Este trabajo presenta una experiencia real de reingeniería de sistemas legados. La misma tuvo lugar en la Administración General de Vialidad Provincial (Santa Cruz) y demandó aproximadamente un año de trabajo. El proyecto de reingeniería tuvo dos objetivos muy concretos: a) desarrollar un proceso de reingeniería que se adecuara los sistemas legados de la organización con la finalidad de poder ser replicado a otros sistemas legados de la misma. Este objetivo dio por resultado la elaboración de una guía para su posterior reutilización a estos contextos. b) su aplicación a uno de los sistemas legados de la organización.

La guía de reingeniería que se expone es un marco de trabajo, un conjunto de lineamiento que ordenan y organizan tareas, actividades y artefactos, con el objeto de administrar y conducir el proceso de migrar estos sistemas. Teniendo en cuenta que ha sido elaborada y usada en las siguientes condiciones: inexistencia de documentación alguna de los sistemas, ausencia del 30% de los códigos fuentes, diferencia entre el lenguaje de programación del actual y futuro sistema, como también en el soporte al modelo de datos.

Este artículo tiene la siguiente estructura: en la Sección II se exponen las motivaciones del trabajo, en la Sección III se describe el proyecto de reingeniería, en la Sección IV se presenta la Guía, en la Sección V su aplicación al sistema de Patrimonio, en la Sección VI se presentan algunos trabajos relacionados y finalmente las conclusiones y trabajos futuros.

II. MOTIVACIONES

La Administración General de Vialidad Provincial (AGVP) de la Pcia. de Santa Cruz, está facultada a administrar e invertir los fondos afectados para el estudio, trazado, construcción, mejoramiento y conservación de caminos y obras anexas en la red de caminos de su jurisdicción. Las tareas y funciones operativas, técnicas y administrativas se llevan a cabo en toda la pcia., a través de 7 distritos y la sede central, en los que se divide y/o compone la estructura. Los distritos se asientan en las localidades de Río Gallegos, Cmdte. Luis Piedra Buena, Puerto Deseado, Las Heras, Perito Moreno, Gdor. Gregores y El Calafate y la Casa Central, sita en la ciudad de Río Gallegos.

Se cuenta con diversas aplicaciones desarrolladas por la Dirección de Informática (Casa Central), que permiten el cumplimiento de sus funciones tales como Patrimonio (S1), Contabilidad (S2), Pago a proveedores (S3), Complementarias de sueldos (S4), Ordenes de Trabajo SC (S5), Viáticos SC (S6), Personal SC (S7) Asistencia SC (S8), Depósito SC (S9), Expedientes (S10), Viáticos Distritos (S11), Sueldos (S12), Judiciales (S13), Certificados (S14), Infracciones (S15), Convenios (S16) y Cargas (S17).

El primer sistema que se estudio fue el de Patrimonio (S1) y se identificaron las siguientes deficiencias y problemas de mantenimiento:

- Portabilibilidad: El sistema se ejecutaba originalmente bajo el sistema operativo D.O.S., por lo cual y conforme a la evolución de los sistemas operativos, el sistema de Patrimonio solo es compatible con la versión de Windows 98. Debido a que fue la última versión de Windows que posibilita la configuración manual de la memoria extendida, imprescindible para su funcionamiento. (D1)
- Impresión de formularios y reportes restringidos a impresoras de matriz de punto. (D2)
- Es un sistema mono-usuario que es utilizado en distintos puntos geográficos (distritos) de la pcia. y la actualización de sus datos son centralizados en el distrito central (Río Gallegos) utilizando diskette 3¼ o CD, lo que no permite contar con la información en tiempo real. (D3)
- Seguridad: No existe ningún esquema de seguridad en el sistema mediante autorización y autentificación de usuarios, lo cual se agrava dado que los archivos de datos (DBF) son fácilmente accesibles y modificables con herramientas tales como Excel.(D4)
- Falta de Código Fuente: Con el transcurso del tiempo y el traspaso de los archivos de un equipo a otro, debido a la actualización del equipamiento informático, varios códigos fuentes se extraviaron quedando solo los archivos compilados. (D5)
- Lenguaje en desuso: el software ha sido desarrollado en un lenguaje y con herramientas en desuso, utilizando técnicas y estilos de programación distintos a los actuales. (D6)
- Código: El código existente presenta varios síntomas indeseables que afectan su legibilidad y por ende su mantenimiento: modularización del código baja; presencia de segmentos de código repetido; uso abusivo de variables globales; designación de nombres de variables y módulos no significativos; bucles no estructurados, etc. (D7)
- Inexistencia de documentación técnica. (D8)
- Inexistencia de documentación operativa del sistema. (D9)

En la Tabla 1 se presenta el análisis que se llevó a cabo sobre los problemas observados (D1 a D9) en el sistema de Patrimonio y su existencia en los restantes sistemas de la organización (S2 a S17). Dicho análisis indica que la reingeniería debe ser aplicada a varios sistemas y por ende el proyecto debe comprender dos aspectos, la elaboración de un marco de trabajo adecuado y específico que pueda ser replicado, y su aplicación a cada una de las instancias (sistemas).

La decisión de migrar, aplicando una reingeniería los sistemas de AGVP, en lugar de desarrollar desde cero las aplicaciones, responden al hecho que los sistemas cuentan con la funcionalidad requerida para realizar las tareas de las áreas donde se encuentran instalados, cumpliendo además con la mayoría de los requerimientos de los usuarios. Debido al tiempo transcurrido desde su implementación, se encuentran suficientemente probados y los usuarios perfectamente adaptados a los mismos, pero las nuevas tecnologías de hardware y software obligan a realizar una reconstrucción de dichas aplicación como así también de los datos.

Se ha indicado al lenguaje de programación en desuso como una deficiencia, específicamente los sistemas señalados de AGVP fueron codificados con herramientas xBase. Las aplicaciones que responden al modelo conocido como xBase (DBase III+, DBase IV, FoxPro para DOS/Window, Clipper, etc.) tuvieron gran auge en las décadas de los 80 y 90.

TABLA I. ANÁLISIS DE DEFICIENCIAS EN LOS SISTEMAS DE AGVP

	D1	D2	D3	D4	D5	D6	D7	D8	D9
S1	X	X	X	X	X	X	X	X	X
S2	X	X	X	X	X	X	X	X	X
S3	X	X	X	X	X	X	X	X	X
S4	X	X	X	X	X	X	X	X	X
S5	-	-	X	X	-	X	X	X	X
S6	-	-	X	X	-	X	X	X	-
S7	-	-	X	X	-	X	X	X	X
S8	-	-	X	X	-	X	X	X	X
S9	-	-	X	X	-	X	X	-	X
S10	-	-	X	X	-	X	X	X	X
S11	-	-	X	X	-	X	X	-	X
S12	-	-	X	X	-	X	X	X	X
S13	-	-	X	X	-	-	-	X	X
S14	-	-	X	X	-	-	-	X	X
S15	-	-	X	X	-	X	X	X	X
S16	-	-	X	X	-	X	X	X	X
S17	-	-	X	X	-	X	X	X	X

Estas herramientas fueron acuñadas por los programadores, por su facilidad de programación y manejo de ficheros (DBF) en microordenadores (PCs), en contraposición a otras herramientas de programación de la época (C, Cobol, Basic). Sin embargo la irrupción de internet, las intranets, BD relacionales, lenguajes de script, tecnologías OO y/o componentes jaquearon las bondades xBase. La falta de oportunidades está dada principalmente en aspectos como la portabilidad, escalabilidad, seguridad, procesamiento multiusuario, etc.

III. PROYECTO DE REINGENIERIA

Como ilustra la Fig. 1, el proyecto de reingeniería se planteo en dos etapas principales. La primer etapa del proyecto estuvo destina a la recolección de información y análisis de métodos, técnicas y herramientas de ingeniería inversa, reingeniería y migración de software y/o sistemas legados. Así en la Etapa 1 del proyecto, se analizaron los modelos de reingeniería de Sommerville [5], SICUMA [6], el modelo Herradura [7] y el modelo Cíclico [8]. Finalmente se decidió tomar como punto de partida el modelo Cíclico, el cual resulta ser bastante conocido y utilizado, y debido a que ofrecía mayor confiabilidad. Este modelo consta de una secuencia de seis etapas bien estructuradas, que hacen posible practicar la reingeniería de software de manera que genere resultados más concretos. Las seis etapas que define son, (i) Análisis de Inventario, (ii) Reestructuración de documentos, (iii) Ingeniería Inversa; (iv) Reestructuración de código, (v) Reestructuración de datos, y (vi) Ingeniería Directa. Estas etapas se pueden realizar de forma secuencial, pero también es posible que estas se puedan repetir generando un ciclo. Estas características hicieron a este modelo proclive a su aplicación, pero aún así fue necesario adaptarlo a las necesidades particulares. Es por ello, que las etapas fueron analizadas y redefinidas. La etapa (ii) Restructuración de documentos no es aplicable en función que no existe documentación alguna del sistema para modificar

y/o actualizar. De igual forma la etapa (iv) Restructuración del Código tampoco es aplicable, dado que se construirá un nuevo código en un lenguaje de programación totalmente diferente.

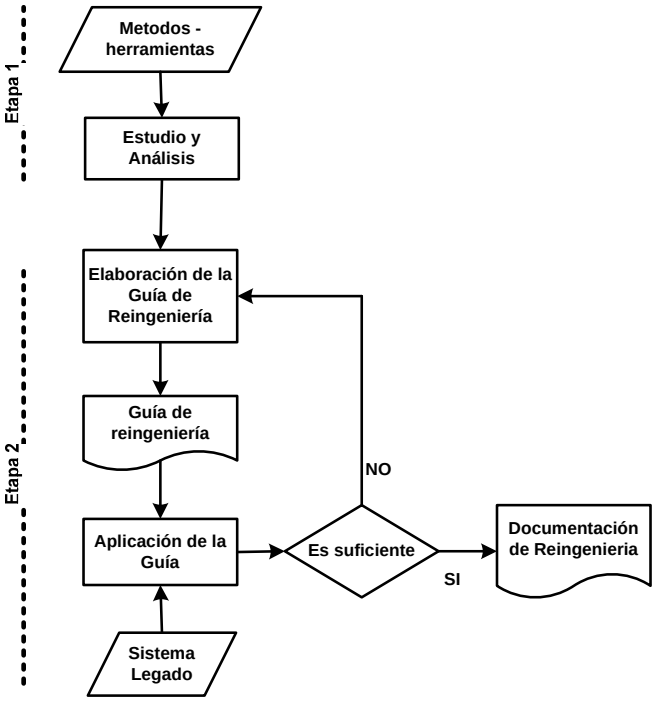


Fig. 1. Esquema del Proyecto de Reingenieria

También se analizaron herramientas que permitieran automatizar alguna de las tareas/actividades. En este sentido, se estudió un conjunto amplio de herramientas conformados por: Rational Rose, Visual Paradigm, Enterprise Architect, Power Disigner, Imagix 4D, Codesurfer, Jeliot, Mogwai ER Designer, SchemaSpy, CASE Studio, ER/Studio, Toad Data Modeler, OllyDbg, Windbg, IDA Pro, Ciasdis, ExDec, Valkyrie Clipper, etc., en los siguientes aspectos: Clasificación (desamblador/decompilador/CASE/depurador/analizador de datos-código, etc.), Tipo de tarea que desempeña, salidas que produce (modelos/diseño/código/informes), Licencia (paga o libre), Lenguaje base de la herramienta, plataforma de ejecución (Linux/Window), entradas que requiere, etc.

La segunda etapa del proyecto consistió en la elaboración de la guía y su aplicación, casi en simultaneo, de manera tal de obtener una retroalimentación que completara la guía. La interacción implicó redefinir las actividades, incorporar más artefactos de documentación, en definitiva probar y mejorar el uso y utilidad de la propuesta.

IV. Guia de Reingenieria

La Guía se ha utilizado para documentar y desarrollar el Sistema de Patrimonio de A.G.V.P. Desde el inicio del proyecto el equipo de trabajo tenia clara convicción que el nuevo sistema a desarrollar sería una aplicación web. Ya que así se superaban las deficiencias y problemas observados. La nueva aplicación debía soportar todas las funciones y operaciones específicas en un entorno multiplataforma (diversas versiones de Windows y Linux), multiusuario, con acceso de distintos puntos geográficos de la pcia., con un entorno gráfico más actual y amigable. Además se buscaba que fuera mantenible, con facilidad para incorporar funcionalidad y con documentación disponible y actualizada.

Como se indicó, al inicio del proyecto, no se contó con planes de proyecto, estimaciones de costos, arquitectura, especificaciones, modelos de requisitos, o diseños. Tampoco existía documentación del usuario. Se disponía solo con el código fuente de la mayoría de los programas fuentes y los archivos de datos o tablas (DBF).

En la Tabla 2 se presenta las Fases de la guía y por cada una de ellas una breve descripción

TABLA II. Guía de Reingeniería

Fase	Objetivo
Análisis de Inventario	Recabar información para obtener una descripción detallada del sistema legado.
Ingeniería Inversa	Analizar los programas (código fuente) con el fin de crear una representación de un nivel de abstracción más elevado que el código fuente. Extraer de los códigos existentes información del diseño arquitectónico y de proceso, e información de los datos.
Reestructuración de Datos	Analizar la arquitectura de datos actual y definir los modelos de datos necesarios.
Ingeniería directa	Generar las nuevas especificaciones
Desarrollo de la nueva Aplicación	Realizar todas aquellas actividades para la implementación del nuevo Sistema

A. *Fases, Actividades y Artefactos*

A continuación se describen brevemente las fases, actividades y artefactos que componen la guía.

Análisis de inventario. Se 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.

Ingeniería Inversa. Es el proceso de análisis de los programas (fuentes y ejecutables) con el fin de crear una representación de un nivel de abstracción más elevado que el código fuente. Se extraerá de los códigos existentes información del diseño arquitectónico y de proceso, e información de los datos. Las actividades que se realizan son:

a) Análisis de la interfaz del Sistema: se interactúa con la todas las interfaces del sistema, el análisis de los menús y los submenús; formularios de datos, reportes generados por el sistema y se confecciona un diagrama de las distintas llamadas.

b) A partir del análisis de interfaz del Sistema, se obtiene un conjunto inicial de requerimientos funcionales y no funcionales del Sistema, el conjunto de entradas y salidas del sistema y un conjunto de entidades u objetos de datos preliminar pero relevante para el sistema.

c) Diagrama de flujo de los programas del Sistema: a partir del programa principal, y con la ayuda de alguna herramienta, se generan los diagramas de flujo de todos los programas. Se analiza el código fuente para obtener información, como las llamadas a funciones, procedimientos y otros programas, tablas utilizadas y acciones sobres las mismas, control de datos, etc.

d) Matriz de llamadas: a partir de la actividad anterior se confecciona una matriz por programa, en la cual se identifican las llamadas a funciones, procedimientos y/o programas.

e) Matriz de tablas: sobre las misma matrices generadas, se incorporan las tablas utilizadas en cada uno de los programas, procedimientos y/o funciones.

f) Matriz análisis de tablas: sobre las matrices de tablas, se identifican las operaciones que se realizan sobre las tablas utilizadas (alta, baja o modificación) por los programas, procedimientos y/o funciones.

g) Listado de archivos de datos potenciales: como resultado del análisis del código fuente y de las matrices resultantes de las actividades anteriores, se identifican los archivos que serian las posibles tablas que almacenan los datos del sistema.

h) Identificación de posibles claves: se analizan los datos de cada tabla y se establecen las posibles claves (primarias y ajenas).

Las actividades mencionadas generan información suficiente que permite construir representaciones y especificaciones de mayor nivel de abstracción como las que se indican en la Tabla 3.

Restructuración de datos. La reestructuración de datos toma como insumos la documentación y registros obtenidos en la etapa anterior: entidades identificadas, diccionarios de almacenamientos, DER, etc. Estos esquemas que denominamos arquitectura de datos actual, aunque no cumplan todos las condiciones de una arquitectura, se analiza minuciosamente y en profundidad. A continuación se define el nuevo modelo de datos necesario, que implica la identificación de los objetos de datos y atributos y, a continuación, se revisan las estructuras de datos a efectos de mejorar su calidad. Estas actividades continúan en la etapa siguiente.

a) Análisis de la estructura de datos: con las actividades precedentes se analiza la estructura de datos existente y se identifican los problemas en dichas estructuras.

b) Creación de nuevas tablas temporales: Se generan la estructura y sus correspondientes tablas que serán utilizados en forma temporal en procesos de adecuación y depuración de las estructuras.

c) Codificación de pequeños programas: se programan pequeños algoritmos para procesar y depurar los datos originales y almacenarlos en las tablas temporales.

Ingeniería directa. Con la información obtenida en las etapas anteriores en esta fase se plantea generar y documentar las nuevas especificaciones del nuevo sistema. También se toman decisiones que tienen fuerte influencia en la posterior implementación e imponen restricciones técnicas. Estas especificaciones refieren principalmente a las siguientes cuestiones:

a) Requisitos funcionales y no funcionales

b) Plan de aseguramiento de calidad

c) Normalización de las tablas heredadas

d) Elección del motor de base de datos

e) Elección de lenguaje de programación

f) Diseño de Interface

g) Definición de clases

TABLA III. GUÍA DE REINGENIERÍA

Actividad/artefacto	Representación (abstracción)
Análisis de la Interfaz del Sistema Recuperación de requerimientos funcionales	Diagrama de Contexto Lista de Eventos Lista de Entradas/salidas Especificación de requerimientos funcionales Casos de Uso
Diagramas de flujo de programas Matrices de llamadas datos y análisis de tablas	Diccionario de Datos Diccionario de Procesos actualización y control-validación) Diccionario de Almacenamientos
Listado de archivos Identificación de claves	Diagrama Entidad Relación

Desarrollo de la nueva aplicación. Dado que se cuenta con las especificaciones funcionales y no funcionales y además decisiones técnicas de implementación tomadas, en esta etapa se procede a realizar tareas de implementación y pruebas.

a) Instalación y configuración de software de base (servidores de aplicaciones y/o de datos).

b) Codificación de interface.

c) Codificación de clases.

d) Codificación de procedimientos almacenados.

e) Diseño y ejecución de pruebas.

V. APLICACION

Como se mencionó la Guía fue aplicada al sistema de Patrimonio, el cual fue desarrollado en año 1991, con el lenguaje FoxPro 2.6. El sistema está compuesto de 71 programas (archivos .prg o .fxp) los que suman un total de 20.787 líneas de código y 12 tablas en las que se almacenan los datos (tablas o ficheros DBF). Dada la extensión de la documentación resulta imposible incluirla completamente en este artículo, por lo cual se presentarán algunos esquemas representativos de cada fase.

La fase de Ingeniería Inversa es clave dado que requiere del examen del sistema existente para recuperar información fundamental. Un primer acercamiento a los requerimientos funcionales soportados y entidades relevantes del sistema se obtienen del análisis de la interfaz. Por ejemplo, la Fig. 3 presenta un submenú del sistema, el que corresponde a "motores" y todas las operaciones relacionadas (opciones), a continuación la representación en un árbol del submenú y el conjunto de requerimientos relacionados, en la Fig. 4. Así mismo de los formularios de carga de datos, se obtienen detalles de la entidad Motor y sus atributos.

Luego se analizan los programas fuentes. Para lo cual debieron ser de-compilados 22 programas de los 71, ante la ausencia de los códigos fuentes correspondientes. Se empleo la herramienta RE-Fox X para esta actividad. Por cada uno se obtiene un diagrama de flujo que permite más fácilmente comprender el flujo de control, identificar los archivos de datos y los campos utilizados.

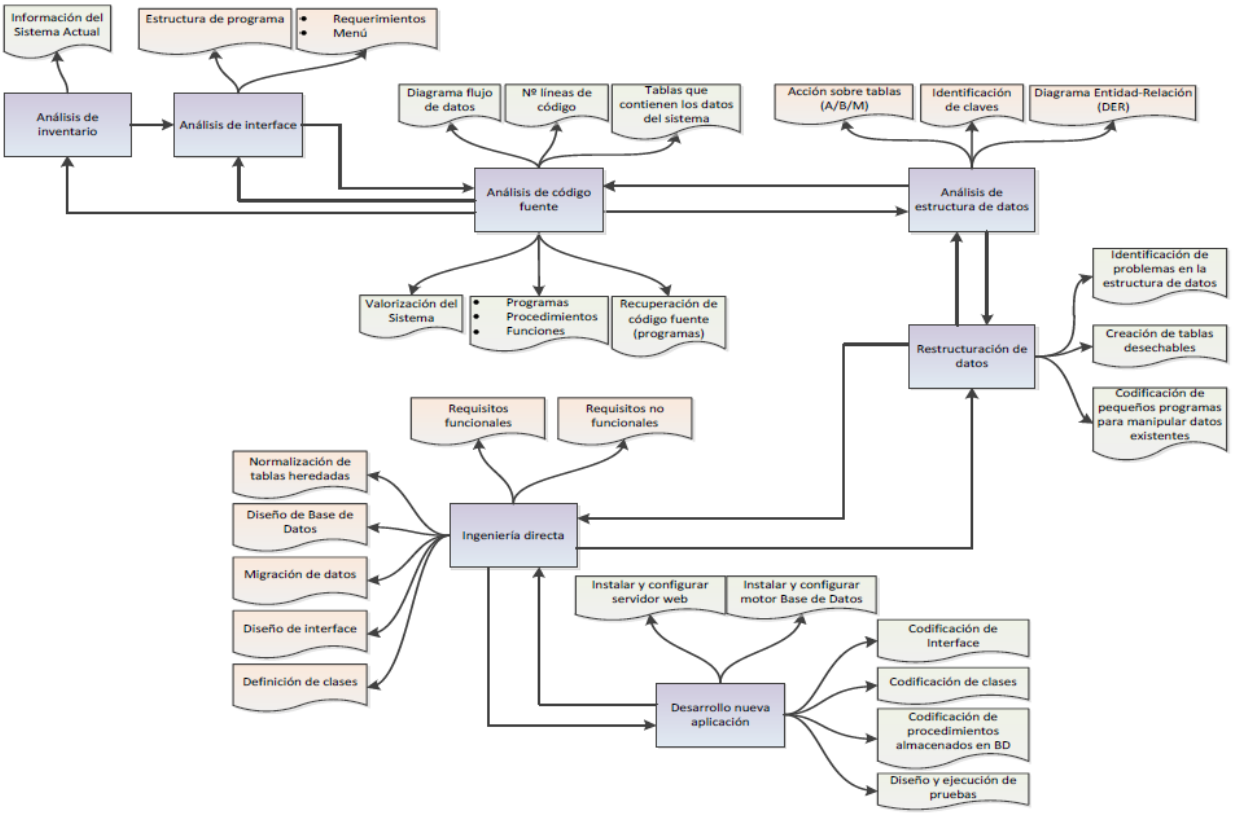
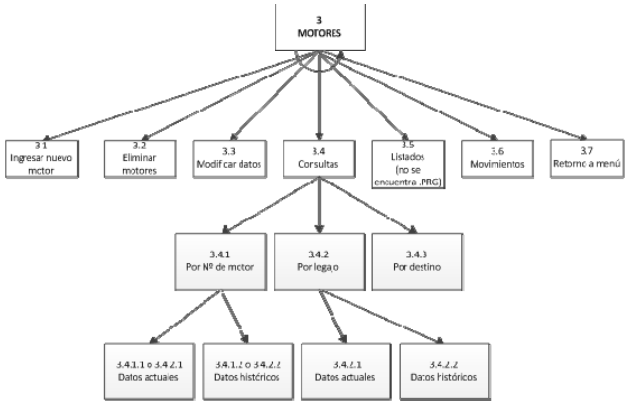


Fig. 2. Guía de Reingenieria: etapas, actividades y artefactos

Fig. 3. Submenú de Motores

Los diagrama de flujo de datos se generaron automáticamente con la herramienta Visustin v7.04, cuyos diagramas aportan más información que la versión clásica de este tipo de diagramas. Dado que los archivos de datos son accedidos por distintos programas/procedimientos y con distintos objetivos (lectura/escritura), se procede a realizar un conjunto de matrices que tiene por objeto identificar los procesos de almacenamiento, procesos de validación, almacenamientos, flujos de datos. La Fig. 5 presenta la matriz correspondiente al programa BAJA.prg.

Aunque esta actividad demanda un gran esfuerzo, permite identificar segmentos de código repetido, operaciones de validación, reglas de negocio implícitas, datos relevantes, etc. Obtenidas las matrices correspondientes a todos los programas, es posible listar los archivos de datos que usa el sistema. Por cada una de estas tablas, se obtiene su estructura y se analiza el uso de cada uno de los campos por los programas y se realiza una verificación con las entidades identificadas en el análisis previo de la interfaz. Luego es posible realizar el Diagrama entidad-relación (DER) del sistema de Patrimonio (Fig. 6).



Requerimientos Funcionales	
1	Realizar alta/baja/modificaciones de datos de los motores.
2	Efectuar movimientos (cambio de automotor).
3	Generar consultas por número de motor o por legajo, estos con la opción de mostrar datos históricos o solo datos actuales.
4	Generar consulta por destino.

Entidad	Motor
Atributos	Nro Motor Cuenta Subcuenta Estado Tipo Documento Nro mm/aa Legajo Original Destino Responsable Legajo Actual Observaciones

Fig. 4. Requerimientos funcionales y Entidades obtenidas del submenú y formularios

BAJAEL.PRG (C:\PATRIMO)						
Tablas utilizadas	Tablas actualizadas	Programas llamados	Procedi-mientos	A	B	M
nomencla			pantalla			
bienes	bienes		mentop	x	x	
legajo			cartel			
areas			codigo			
peralfa			mensaje			
historia	historia		siono		x	x
borratin	borratin			x		x

Fig. 5. Matriz del programa BAJA.prg

La restructuración de datos, implicó analizar las tablas y sus campos, además de los usos de los mismos en los programas. Se identifican distintos tipos de problemas:

- Distintas entidades almacenadas en el mismo archivo (se usan campos de estado para su diferenciación)
- Diseño deficiente por falta de normalización
- Definiciones de campos innecesarios (no utilizados)
- Claves no declaradas
- Estructuras ocultas no declaradas (por ejemplo es el caso de la numeración de documentos para la cual se almacena el número del documento junto con el año en un campo de tipo carácter).

Se procede a una depuración que sanee estas inconsistencias. Se generan las nuevas estructuras y la migración se lleva a cabo por medio de programas codificados a tal fin. Estas actividades deben realizarse con extrema precaución dado que existe un volumen de datos que debe ser preservado y estar disponible en el futuro sistema. A medida que se realiza la migración de cada estructura se hacen actividades de chequeo que garantizan la total y correcta migración, como llevar un registro minucioso.

En la fase de Ingeniería Directa, no solamente se recupera la información de diseño del antiguo sistema de Patrimonio, además, utiliza esta información en un esfuerzo por mejorar su calidad global. En la mayoría de los casos, el software procedente de una reingeniería vuelve a implementar la funcionalidad del sistema existente, y añade nuevas funciones y/o mejora el rendimiento global.

Se inicia desarrollando la nueva especificación de requerimientos, la cual incluye en principio los funcionales y fundamentalmente se definen y especifican los requerimientos no funcionales, (inexistentes en el sistema legado) los cuales se identifican y jerarquizan en:

- Atributos de calidad: desempeño, escalabilidad, facilidad de uso e ingreso de información, mantenimiento, operatividad, seguridad, validación de la información
- Arquitectura
- Respaldos
- Sistema de flujo de datos.

Se elabora el plan de aseguramiento de calidad que propone estándares (objetivos e indicadores) a cumplir en cuanto a tres factores: entrega del producto, interfaz de usuario y rendimiento del sistema. Los dos últimos factores implican un cambio importante para el usuario en su operatoria diaria con el futuro sistema.

Se finaliza con el diseño de los datos por lo cual se realiza la normalización de los datos y el diseño de la nueva base de datos.

Se elabora el diseño de la interfaz, el cual define el layout de pantallas, tipo y estructura de menúes, estructura de formularios, mensajes y otros componentes gráficos.

En la fase de Desarrollo de la Nueva Aplicación se realizan actividades que finalizan el diseño, pero principalmente se procede a la implementación de la funcionalidad y pruebas.

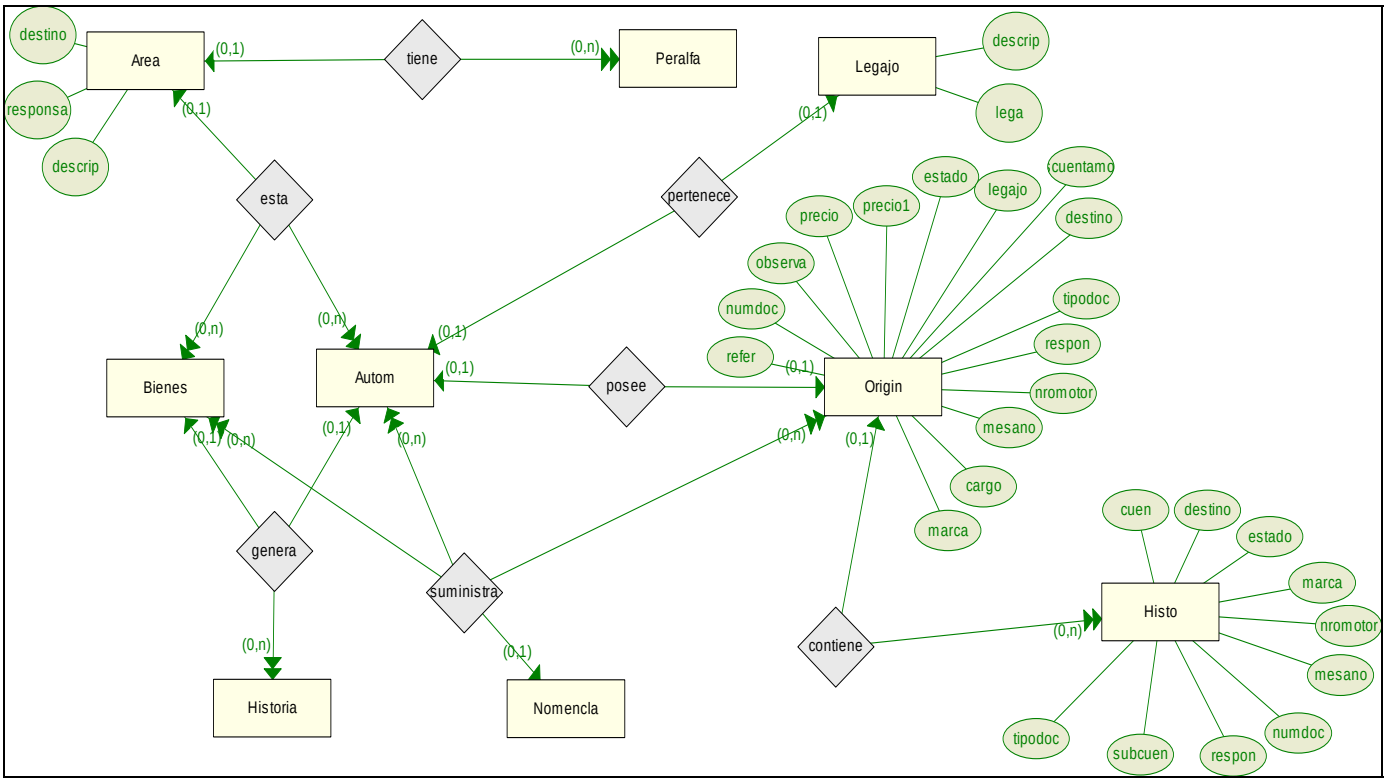


Fig. 6. ER del Sistema de Patrimonio

Dado que se ha decidido que el nuevo sistema será una aplicación web, se decide utilizar herramientas Microsoft, SQL Server Express, SQL Management Studio Express, ya que dichos entornos permiten a los desarrolladores combinar un amplio grupo de herramientas gráficas y editores de scripts que proporcionan acceso a SQL Server Express, a los programadores y administradores de todos los niveles.

Se desarrollan las clases del sistema, lo cual no reviste dificultad en este punto del proyecto, la mayoría de las entidades están identificadas (Bienes, motores, nomenclador, etc.) y se añaden principalmente aquellas clases que tendrán las responsabilidades de gestión de las mismas. Se completan especificación de las mismas y sus relaciones.

Las actividades de diseño y ejecución de pruebas están orientadas a realizar pruebas de unidad, integración y del sistema con el objetivo de probar los requerimientos planteados. En la Tabla 4 se presenta por cada tipo de prueba algunas de las acciones llevadas a cabo.

TABLA IV. PRUEBAS Y ACCIONES

Pruebas	Acciones
Integridad	Verificar el acceso al sistema. Verificar la recuperación correcta de las modificaciones realizadas en la base de datos. Verificar accesos simultáneos de lectura y escritura de datos.
Interfaz de Usuario	Verificar la navegación a través de un conjunto de pantallas. Navegar a través del menú, verificando que cada interfaz de usuario se comprenda fácilmente.
Técnicas	Comprobar que los procedimientos almacenados y métodos de acceso a la base de datos funcionan correctamente. Inspeccionar la base de datos para asegurar que los datos son los correctos, que todos los eventos de la base de datos ocurren adecuadamente. Asegurar la navegación correcta entre formularios de la aplicación, la entrada, el proceso y recuperación de los datos. Realizar operaciones posibles en cada formulario con entradas válidas e inválidas para verificar los resultados esperados, mensajes de error o advertencias adecuadas. Crear y/o modificar pruebas para cada una de las ventanas.

Al finalizar esta etapa se cuenta con la nueva aplicación desarrollada, los datos migrados y la documentación técnica actualizada y disponible.

VI. TRABAJOS RELACIONADOS

Varios esfuerzos de reingeniería han sido reportados en la literatura. En [9] se describe un enfoque para la construcción de componentes reutilizables de grandes sistemas existentes mediante la identificación de los subsistemas estrechamente acopladas contenidos en el sistema. El enfoque se basa principalmente en el uso de métodos informales y experimentales y el objetivo principal era hacer observaciones acerca de las experiencias vividas con algunos sistemas grandes. Lewis y McConnell [10] describen un proceso de reingeniería y su aplicación a un sistema embebido en tiempo real. El proceso de siete pasos describe hitos de alto nivel que van desde la ingeniería inversa y la reingeniería de reorientación y la prueba final. En [11] se presenta un caso de estudio que ilustra cómo una combinación de técnicas de análisis orientado a objetos y estructurado y diseño estructurado se pueden utilizar en conjunto para rediseñar una aplicación existente que es utilizada por Texas Instruments para analizar el tráfico en redes de área local. En [12] se presenta una experiencia de reingeniería de una aplicación científica heredada que resulta obsoleta en cuanto a operatividad, aspecto y software de base sobre el que se ejecuta, pero de probada eficiencia y que mantiene su funcionalidad. Se muestran las características de un proceso de desarrollo que se adapta a este tipo de aplicaciones, verificado mediante el caso de estudio, la transformación de una aplicación escrita en un lenguaje imperativo, no estructurado, a un nuevo lenguaje visual y orientado a objetos, describiendo las diversas fases de la metodología aplicadas a un caso concreto.

VII. CONCLUSIONES

Se ha presentado una guía para la reingeniería de sistemas legados que ha sido puesta en práctica a un sistema real. La elaboración y aplicación prácticamente en simultáneo ha permitido validar su razonabilidad y utilidad. Asimismo no es posible generalizar su aplicabilidad a cualquier sistema legado, acotamos su uso a los sistemas legados que responden a las características señalas. Las actividades y artefactos empleados en cada fase pueden ser ampliados y la principal desventaja que encontramos es que la mayor parte del trabajo ha sido manual. Aunque ha demandado mayor esfuerzo y tiempo ha permitido adquirir experiencia.

El trabajo futuro está dirigido a la reingeniería de los restantes sistemas legados de AGVP, pero antes nos abocaremos a: desarrollar algunas herramientas utilitarias que automaticen algunas de las tareas, por ejemplo la construcción de las matrices, las cuales han resultado ser un artefacto de gran utilidad en nuestra experiencia. Otro objetivo a cumplir es incorporar un modelo de estimación esfuerzo/tiempo, que nos permita planificar la reingeniería de cada sistema.

AGRADECIMIENTOS

El presente proyecto ha sido parcialmente financiado por la Agencia Nacional de Promoción Científica y Tecnológica – MINCyT – Argentina.

REFERENCIAS

[1] K. Bennett, “Legacy Systems: Coping with success”, IEEE Software, vol. 12, no. 1, 1995, pp. 19 – 23.

[2] H. Sneed, “Planning the Re-engineering of Legacy Systems”, IEEE Software, vol. 12, no. 1, 1995, pp. 24 – 34.

[3] B. Lientz y E. Swanson, “Software Maintenance Management”. Addison Wesley, Reading, MA, 1980.

[4] E. J. Byrne, “A Conceptual Foundation for Software Reengineering,” Proceedings de Conference on Software Maintenance, pp. 226–235, IEEE, 1992.

[5] I. Sommerville, “Software Engineering”. 6ta edición. Addison Wesley, 2001.

[6] J. Leiva, “Construcción de especificaciones de interfaces en un proceso de reingeniería,” 2da. Conferencia Iberoamericana en Sistemas, Cibernética e Informática, 2003, USA, pp 202-208.

[7] J. Bergey, D. Smith, N. Weiderman y S. Woods, “Option Analysis for Reengineering (OAR): Issues and Conceptual Approach” (CMU/SEI-99-TN-014). Pittsburg, Pa.: Software Engineering Institute, Carnegie Mellon University, 1999.

[8] R. Pressman, “Ingeniería del software: Un enfoque práctico”. 6ta Edición. McGraw-Hill, 2005.

[9] J. Neighbors, “Finding reusable components in large systems,” *Proceedings of the Third IEEE Working Conference on Reverse Engineering*, pp. 2–10, IEEE, November 1996.

[10] B. Lewis y D. J. McConnell, “Reengineering Real-Time Embedded Software onto a Parallel Processing Platform,” *Proceedings the Third Working Conference on Reverse Engineering*, pp. 11–19, November 1996.

[11] G. Gannod, G. Sudindranath y M. E. Fagnani, “PACKRAT: A Software Reengineering Case Study”. Proceedings of the 5th Working Conference on Reverse Engineering, 1998, pp. 125-134, IEEE.

[12] J. Alvarez Garcia, M. Sanchez M. y M. Moreno Garcia, “Metodología De Reingeniería Del Software Para La Remodelación De Aplicaciones Científicas Heredadas”. Informe Técnico – DPTOIA-IT-2004-003 - 2004 – Repositorio documental de la Universidad de Salamanca

Franco Bianchiotti. Se gradúo de Licenciado en Sistemas en la Universidad Nacional de la Patagonia Austral. Ha sido becario de la Agencia Nacional de Promoción Científica y Tecnológica (MINCyT – Argentina). Actualmente se desempeña en la Dirección de Informática de la Administración General de Vialidad Provincial, desde el año 2007.

Sandra Casas. Es Profesora Asociada de la Universidad Nacional de la Patagonia Austral, (Argentina), dónde además es miembro del Instituto de Tecnología Aplicada (ITA). Se doctoró en la Universidad de Vigo en el año 2008 y su línea de investigación refiere a temáticas relacionadas al desarrollo de software. Dirige el grupo de investigación GISP del ITA y proyectos de investigación desde el año 2005.