Estado del arte de la reingeniería y la ingeniería inversa: 2001-2003

Ignacio García-Rodríguez de Guzmán, Macario Polo, Mario Piattini

Grupo ALARCOS
Dpto. Informática – Universidad de Castilla-La Mancha
igarcia@proyectos.inf-cr.uclm.es

Abstract. En este artículo, se presenta el estado del arte sobre reingeniería e ingeniería inversa en los últimos años, basado en la investigación publicada en las tres principales conferencias sobre estos temas durante los años 2001, 2002 y 2003. A través de este estudio bibliográfico, se han analizado datos referentes a la proporción de trabajos que tratan sobre la ingeniería inversa frente a los que versan sobre la reingeniería, la importancia que tiene la automatización de estos procesos, que tipo de sistemas heredados requieren el uso de estas técnicas con más frecuencia y, entre otras cosas, cuales son las carencias o campos de la ingeniería del software donde la reingeniería y la ingeniería inversa han profundizado menos a día de hoy.

1.- INTRODUCCIÓN

En los últimos años, la comunidad científica internacional ha reconocido la importancia -especialmente económica - que tiene el mantenimiento del software. Hace ya años que diversos autores vienen reclamando la necesidad de prestar más atención al mantenimiento del software (Pigoski 1997), basándose en que todos los estudios demuestran que es la etapa del ciclo de vida de un producto software que más recursos consume (Piattini et al. 2000). Afortunadamente, la comunidad científica ha escuchado este llamamiento y, en los últimos años, han surgido iniciativas, reuniones, publicaciones, etc., dedicadas a abordar este asunto.

Así, por ejemplo, en la conferencia ICSE celebrada en el año 2000 en Limerick (Irlanda), se incluyó una sesión dedicada al "futuro de la ingeniería del software" en la cual, Keith Bennett y Vaclav Rajlich, dos conocidos autores especializados en mantenimiento y evolución del software, presentaron una "hoja o mapa de ruta" (*roadmap*) (Bennet et al. 2000) sobre mantenimiento de software que pretendía realizar una llamada de atención a la comunidad investigadora. Dentro de esta propuesta se incluían el desarrollo de mejores métodos y técnicas de ingeniería inversa y reingeniería.

Como sabemos, la ingeniería inversa y, en particular, la recuperación de la arquitectura, ayuda a la extracción de representaciones de más alto nivel de sistemas heredados, y ofrece soporte a los ingenieros para la evaluación, el mantenimiento y el desarrollo de sistemas software a gran escala. A fin de producir estas vistas arquitectónicas, las herramientas de ingeniería inversa son capaces de procesar

distintos tipos de artefactos software disponibles tales y como pueden ser el código fuente, escenarios de funcionamiento, documentación, información de dominio y conocimiento del experto (Chikofsky et al. 1990).

Por otra parte, de acuerdo con (Arnold 1992), la reingeniería es el proceso de aplicar en primer lugar, ingeniería inversa a un sistema software existente para obtener estructuras de mayor nivel de abstracción para, en segundo lugar, aplicar una fase de ingeniería directa mediante la cual se obtenga una nueva versión del sistema con nuevas funcionalidades o de menor calidad. De esta forma, en la primera fase de la reingeniería obtendríamos una serie de especificaciones abstractas del sistema heredado (*legacy system*) necesarias para construir nuevas implementaciones del sistema software.

Dentro del proceso de reingeniería, las técnicas de ingeniería inversa son utilizadas para la comprensión de la estructura y el comportamiento de un sistema software del que no se dispone documentación, o que caso de disponerse está obsoleta o es insuficiente (Briand et al. 2003). En muchas ocasiones, el proceso de ingeniería inversa es llevado a cabo a través de un *extractor*, que analiza sintácticamente el código para extraerlos datos del código (Lin et al. 2003), y son estos extractores los que implementan las distintas técnicas que actual y tradicionalmente se han utilizado para la ingeniería inversa.

Mediante el proceso de reingeniería, y dentro de este, en la ingeniería inversa, no se pretende exclusivamente recuperar especificaciones a más alto nivel del sistema heredado. Además de este tipo de información, también se pretende obtener documentación, información que resulta importante a la hora de analizar este tipo de sistemas software y mejorar su comprensión y mantenimiento (Riva et al. 2002c). Esta información, en muchas ocasiones, bien se ha perdido con el tiempo o incluso es inexistente, como ocurre con muchos sistemas Web actuales, que debido a ciclos de desarrollo rápidos merman el desarrollo de documentación asociada, lo que hace que su posterior etapa de mantenimiento sea más difícil y costosa (Di Lucca et al. 2001b).

Este artículo analiza las principales publicaciones presentadas en congresos durante los tres años transcurridos desde la propuesta de Bennet y Rajlich (2000), con el fin de conocer en qué estado se encuentra la investigación en ingeniería inversa y reingeniería.

2. ANÁLISIS DE LAS PRINCIPALES CONFERENCIAS

Para la realización de este estudio, se ha recurrido a la consulta de publicaciones especializadas en la reingeniería y la ingeniería inversa. En concreto se han analizado todos los artículos publicados en las siguientes conferencias: “Working Conference on Reverse Engineering” (WCRE), “Conference on Software Maintenance and Reengineering” (CSMR) e “International Conference on Software Maintenance” (ICSM) organizadas por la IEEE Computer Society y que son las de mayor impacto en este área.

El número de trabajos que se han analizado se resume en la Table 1, clasificados según la conferencia y el año de celebración, sumando un total de 422.

Table 1. Total de trabajos analizados para el estudio

Fuente\año	2001	2002	2003	Total
ICSM	87	78	56	221
CSMR	27	28	40	95
WCRE	38	33	35	106
Total	152	139	131	422

Una vez se dispuso de todos los trabajos que formaban parte de los citados congresos en sus correspondientes ediciones (utilizando la biblioteca digital de la IEEE Computer Society), se realizó una selección de los posibles candidatos a ser considerados, siempre atendiendo a diferentes criterios:

- Existencia de palabras clave en el artículo, tales como pueden ser *"reverse engineering"* y "*reengineering"*.
- Idea desprendida del título del trabajo.
- Contenido del *abstract*.
- En casos dudosos, la lectura de algunas secciones o puntos clave.

En la Table 2, se presentan los artículos seleccionados utilizando estos criterios y que representan un 11% del total de artículos aceptados en estas conferencias.

Table 2. Número de trabajos relacionados con la reingeniería y la ingeniería inversa

Fuente\año	2001		2002		2003	
	Total	**Rev/Ree**	**Total**	**Rev/Ree**	**Total**	**Rev/Ree**
ICSM	87	4	78	3	56	5
CSMR	27	3	28	3	40	3
WCRE	38	8	33	5	35	6
Total	152	15	139	11	131	14

3.- MARCO DE CLASIFICACIÓN

Con el fin de clasificar los trabajos analizados, se ha elaborado un marco (véase Table 3) que consta de los siguientes apartados:

- ***Ref.***: Referencia bibliográfica del trabajo.
- ***Entorno de partida***: Tanto en la ingeniería inversa como en la reingeniería, el punto de inicio es un sistema heredado. Puesto que los sistemas heredados pueden tener naturalezas muy diversas consideramos como un dato

importante el entorno de inicio, al que se aplica el proceso de recuperación de información, p. ej. COBOL, C, etc.

- ***Entorno objetivo***: Al que se llega tras la aplicación de la reingeniería y la ingeniería inversa. Puede ser bien un nuevo sistema mejorado o con nuevas características (cuando es reingeniería) o un conjunto de especificaciones a niveles de mayor abstracción a partir de los que trabajar (ingeniería inversa).
- ***Propósito***: Intenta resumir la intención de los autores del trabajo.
- ***Herramienta:*** Indica si los autores proponen una herramienta que implemente su propuesta.
- ***Rev/Ree***: Expresa si el artículo trata sobre ingeniería inversa (*Rev*) o reingeniería (*Ree*).

En la Table 3 se presentan los trabajos analizados utilizando este marco. En algunos casos, no es posible cumplimentar todos los apartados debido a que la información publicada en los artículos no siempre es todo lo exhaustiva que cabría esperar.

Table 3. Trabajos relevantes

Ref	Entorno de partida	Entorno objetivo	Propósito	Her	Rev/ Ree	Método
(Sartipi et al. 2003)	Lenguajes estructurados	Mayor nivel de representación	Un conjunto de grafos para representar la información de diferentes formas	Sí	Rev	Grafos
(Mitchell 2003b)	Sistemas software (Énfasis en sistemas grandes)	Sistema software representado en *clusters*	Una herramienta automática de *clustering* llamada Bunch. Un conjunto de algoritmos de *clustering* metaheurísticos.	Sí	Rev	Algoritmos de clústering
(Sartipi 2003)	Sistemas software desarrollados en C o Pascal	Arquitectura del sistema software	A través de patrones arquitectónicos y técnicas de correspondencia en grafos, este método trata de recuperar la arquitectura de los sistemas heredados.	Sí	Rev	Patrones, Grafos
(Balanyi et al. 2003)	Código fuente en C++	Patrones de diseño empleados y las clases de cada patrón.	DPML, un language basado en XML, para representar, crear o modificar patrones. Entorno de ingeniería inversa llamado Columbus	Sí	Rev	Patrones. Grafos
(Johnson et al. 2003)	Dos o más sistemas de información heredados.	Un modelo que permita la cooperación de estos.	*JR-Methodology*. Mediante ingeniería inversa, esta metodología consigue un modelo *K* que es la base para la interoperabilidad entre los sistemas de información. Se obtiene una comunicación *Half-duplex* entre los sistemas.	No	Rev	Métodos matemáticos
(Ferenc et al. 2002)	Proyectos en C++	Diferentes tipos de representación a alto nivel del proyecto.	Un *framework* que integra diferentes tipos de tareas de la ingeniería inversa, proveyendo además una única interfaz.	Sí	Rev	Grafos
(Tonella et al. 2002)	Sistemas software (probado en sistemas escritos en C++)	Un grafo de flujo de objetos, obtenido de la especificación del sistema.	Algoritmo de análisis estático para la extracción de diagramas de objetos a partir del código Dos técnicas para la recuperación del diagrama de objetos, la primera que explota la información estática, y la segunda que considera un conjunto de trazas de ejecución asociadas con los casos de prueba disponibles para un programa.	No	Rev	Análisis estático, Grafos.

(Stroulia et al. 2002)	**Sistemas heredados**	**Conjunto de servicios obtenidos del sistema heredado inicial mediante reingeniería.**	**Mediante un conjunto de métodos (*CelLEST*), se realiza el proceso de reingeniería consiguiendo modelar la interfaz mediante diagramas de transición de estados, además través de los patrones de interacción, recuperar las especificaciones de las funciones de la aplicación y construir nuevos *front-end* para hacer la aplicación accesible vía Web.**	**Sí**	**Ree**	**Interacción entre el usuario y el sistema heredado**
(Bianchi et al. 2001)	**Sistemas heredados escritos en lenguajes procedimentales.**	**El mismo sistema, pero mejorado.**	**La reingeniería se lleva a cabo de manera gradual sobre los componentes del sistema, sin que el sistema tenga que ser detenido o duplicado.**	**No**	**Ree**	**No especificado**
(Eisenbarth et al. 2001)	**-**	**-**	**Mediante el comportamiento dinámico y diversas partes del código, se genera información de manera automática.**	**Sí**	**Rev**	**A. estático y dinámico. Grafos**
(Sartipi et al. 2001)	**Sistema heredado.**	**Arquitectura del sistema heredado.**	**Se propone un *framework* para la recuperación de la arquitectura basándose en técnicas de coincidencia inexactas.**	**Sí**	**Rev**	**Patrones, Grafos**
(Tonella et al. 2001)	**Código fuente de un sistema heredado.**	**Conjunto de diagramas en UML.**	**Algoritmo para la mejora de la precisión de los diagramas de clases UML extraídos del código.**	**Sí**	**Rev**	**Alg. basados en grafos**
(Draheim et al. 2003)	**Sistemas basados en JSP**	**Documentación sobre la interfaz del sistema.**	**Este entorno se usa para detectar los errores potenciales relacionados con la seguridad y otros aspectos.**	**.Sí**	**Rev**	**Gramáticas**
(Saeed et al. 2003)	**Versión 3.2.3. de XFig (Aplicación escrita en C)**	**Conjunto de clusters de software.**	**Se abordan conceptos de *clustering* del software y de mantenimiento del software. El *clustering* del software se utiliza para la ingeniería inversa y la re-modularización.**	**No**	**Rev**	**Técnicas de clústering.**
(Villavicencio 2003)	**Variables de salida del código**	**Especificaciones formales**	**Aplicación del slicing condicional al RPC (*Reverse Program Calculation*) para generar representaciones a mayor nivel de abstracción.**	**No**	**Rev**	**Mét. matem., *Slicing* condicional.**
(Riva et al. 2002a)	**-**	**-**	**Técnica que combina el análisis estático y dinámico de la información de la arquitectura para la reconstrucción de la misma. Podemos crear distintas vistas de la arquitectura útiles para la descripción del sistema.**	**Sí**	**Rev**	**A. estático y dinámico, grafos.**
(Yeh et al. 2002)	**Sistema heredado orientado a objetos.**	**Conjunto de relaciones de agregación.**	**Reingeniería sobre relaciones de agregación basándose en la propagación de las operaciones.**	**No**	**Rev**	**Método basado en la propagación.**

(Di Lucca et al. 2002)	**Aplicaciones Web**	**Diagramas UML que describan distintas vistas de la aplicación.**	**Los diagramas UML se utilizan para modelar un conjunto de vistas que describen distintos aspectos de las aplicaciones Web a diferentes niveles de abstracción. Proponen WARE, una herramienta para hacer ingeniería inversa.**	**Sí**	**Rev**	**No especificado**
(Wilde et al. 2001)	**CONVERT3, un sistema basado en FORTRAN**	**Ubicación en el código de una característica concreta.**	**Dos métodos para detectar características en el código fuente. Los métodos son "*Source Reconnaissance technique*" y "*Dependency Graph search method*".**	**Sí**	**Rev**	**Grafos de dependencia**
(Martin et al. 2001)	**Código fuente en C**	**Código fuente en Java**	**Método para transformar código en C a código en Java. Las herramientas propuestas son C2J para pasar de C a Java y C2J++, que pasa C++ a clases Java**	**Sí**	**Ree**	**No especificado**
(Henrard et al. 2001)	**Bases de Datos**	**-**	**Se propone utilizar dentro de la metodología DBRE para ing. inv. en BD diferentes técnicas para la obtención de dependencias de datos. "V*ariable dependency graph*", "*system dependency graph*" y "*program slicing*".**	**Sí**	**Rev**	**Modelos de reflexión jerárquica.**
(Koschke et al. 2003)	**-**	**-**	**En este traba jo se extiende la modelo de reflexión original al modelo de arquitectura jerárquica (propuesto por (Koschke et al. 2003)), describiendo también como aplicar esta técnica.**	**No**	**Rev**	**Modelos de reflexión jerárquica**
(Lin et al. 2003)	**Código fuente en C/C++**	**Grafo ASG**	**Método para validar la completitud de la semántica de un extractor. Además, se detallan cuatro niveles de completitud incremental sobre los informes extraídos del código fuente.**	**Sí**	**Rev**	**Grafos (ASG)**
(Briand et al. 2003)	**Trazas de la ejecución de un programa**	**Diagramas de secuencia en UML**	**Extraer mediante ingeniería inversa diagramas de secuencia UML de las trazas de ejecución. Se propone una metodología y se valida mediante un caso de estudio.**	**No**	**Rev**	**A. din., reglas de transf., metamodelos**
(Beyer et al. 2003)	**ASG (Abstract Syntax Graph)**	**Informes y patrones identificados**	**Una herramienta para recuperar información mediante consultas sobre un grafo de representación del código.**	**Sí**	**Rev**	**Grafos**
(Alvaro et al. 2003)	**Sistema heredado**	**Diseño original del sistema**	**Entorno de reingeniería basado en componentes para la reconstrucción de sistemas heredados.**	**Sí**	**Ree**	**No especificado.**
(Knodel et al. 2003)	**Entornos basados en componentes**	**Informes y patrones identificados**	**Una aproximación para la extracción genérica de información para sistemas basados en *frameworks* mediante la combinación de análisis léxico con búsqueda de patrones para obtener la especificación partir del código fuente.**	**No**	**Rev**	**Proceso de análisis, patrones**
(Schwarz et al. 2002)	**Código máquina**	**Instrucciones originales**	**Los autores examinan dos algoritmos para la extracción de instrucciones del código máquina, y proponen uno nuevo desarrollado como híbrido de los dos anteriores.**	**No**	**Rev**	**Métodos para desensamblar el código**

(Stoermer et al. 2002)	**-**	**-**	**Introducción a los patrones para la reconstrucción de la arquitectura.**	**No**	**Rev**	**Técnicas basadas en patrones.**
(Pinzger et al. 2002)	**Java**	**Aspectos arquitectónicos**	**Un extractor de modelos a partir del código fuente que combina el análisis léxico con elsintáctico. Utiliza un lenguaje para patrones con distintos niveles de granularidad.**	**Sí**	**Rev**	**Análisis léxico y sintáctico, patrones**
(Henrad et al. 2002)	**-**	**-**	**Seis estrategias que proveen diferentes niveles de calidad e inducen diferentes costes en la migración de aplicaciones que hacen uso intensivo de los datos.**	**No**	**Ree**	**Conversión de BDs, *wrappers*.**
(Riva et al. 2002b)	**Código fuente de una aplicación**	**Documentación organizada en distintos niveles de abstracción.**	**Método para la generación de documentación de un sistema heredado. Consiste en dos fases. La primera para la reconstrucción de la arquitectura y la segunda para la re-documentación de la arquitectura.**	**Sí**	**Rev**	**Grafos (GXL)**
(Blaha 2001a; Blaha 2001b)	**Bases de datos**	**Representación a varios niveles de abstracción**	**Compilación de los resultados obtenidos de la ingeniería inversa aplicada sobre 35 bases de datos.**	**No**	**Rev**	**No especificado**
(Thiran et al. 2001)	**Sistemas heredados (COBOL, BD)**	**Conjunto de *wrappers***	**Un conjunto de *wrappers*, y arquitectura genética que permite instanciar diferentes modelos de datos y sistemas.**	**Sí**	**Rev**	***Wrappersl***
(Mancoridis et al. 2001)	**Código en C, C++ y Java**	**-**	**Portal-web para la ingeniería inversa de sistemas software.**	**Sí**	**Rev**	**No especificado**
(Di Lucca et al. 2001a)	**Aplicaciones Web**	**Modelos UML**	**Se pretende solucionar la falta de documentación provocada por los cidos rápidos diseño y desarrollos de las aplicaciones Web.**	**Sí**	**Rev**	**A. estático, dinámico.**
(Vanderdonckt et al. 2001)	**Páginas web**	**-**	**Recuperar un modelo de presentación de una página web de acuerdo a múltiples opciones en la ingeniería inversa..**	**Sí**	**Rev**	**A. estático del código HTML**
(Bychkov et al. 2001)	**Sistemas heredados como bases de datos**	**Entorno "*Net-centric*".**	**Se propone un conjunto de herramientas para ayudar al ingeniero a realizar la migración de sistemas heredados a entornos "*Net-centric*"**	**Yes**	**Ree**	**Grafos (GXL), *Wrappers***
(Alhajj et al. 2001)	**Base de datos relacional**	**Base de datos orientada a objetos.**	**Un método para transformar bases de datos relacionales en bases de datos orientadas a bases de datos. Mediante RGID, desarrollado por los autores, se soporta toda la información contenida en el diccionario.**	**No**	**Ree**	**Grafos(RIDG, *Relational Intermediate Direct Graph*)**

4. ANÁLISIS DE LOS TRABAJOS

Del análisis de los trabajos presentados en el apartado anterior se pueden extraer algunas conclusiones preliminares.

En cuanto al entorno de partida (véase Fig. 1), más del 50% de los trabajos parte de lenguajes estructurados (C o COBOL), lenguajes orientados a objetos (C++ o Java), y bases de datos. No obstante, es también importante destacar la gran cantidad de trabajos en los que no se especificaba la plataforma de origen (37%). Esta considerable porción está compuesta en su mayoría por estudios de carácter teórico, en los cuales los autores, bien diseñaban nuevas estrategias para analizar los sistemas heredados, bien mejoraban algunas de las propuestas existentes, obteniendo algoritmos mejorados o incluso híbridos (Schwarz et al. 2002) a partir de otros debidamente referenciados.

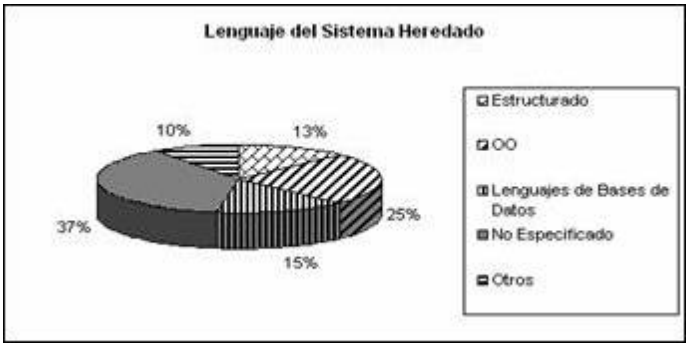

Fig. 1. Proporción de los estudios según el lenguaje del sistema heredado

En la Fig. 2, se resume en términos de porcentajes, los distintos tipos de sistemas heredados sobre los que se centran los trabajos analizados. Los grupos más representativos han sido aplicaciones clásicas, aplicaciones Web, bases de datos y sistemas no especificados. Cabe destacar las aplicaciones clásicas, ya que salta a la vista que una parte considerable de los trabajos abordan la reingeniería o la ingeniería inversa sobre aplicaciones clásicas. Entre otras cosas, esto puede deberse a que actualmente existan todavía en funcionamiento sistemas heredados antiguos que necesiten ser remodelados o mejorados para adaptarse a las nuevas tendencias y necesidades.

En la Fig. 2, nos encontramos que el segundo tipo de sistemas heredados estudiados con más frecuencia son las bases de datos. En este caso, los trabajos son de dos tipos, o bien se trata de la recuperación eficiente de la estructura de la base de datos a partir de los ficheros existentes; o bien, de obtener una representación de más alto nivel (modelos conceptuales) a partir de un esquema relacional. En este último caso, se intenta obtener modelos que representen lo más fielmente la semántica inicial recurriendo al uso de heurísticas y otras técnicas.

Comentaremos también sobre la Fig. 2, la presencia de trabajos que investigan la ingeniería inversa y la reingeniería en aplicaciones Web, que aunque

gozan tan sólo con un 10% de trabajos seleccionados cabe destacar que prácticamente el 100% de los mismos están orientados a la ingeniería inversa. Este dato es importante, ya que de él se deduce que de estos sistemas, sólo se pretende obtener descripciones de alto nivel, y esto se debe a que la demanda actual de sistemas de este tipo provoca que se realicen ciclos de vida cortos para atender a las demandas del mercado. Una de las consecuencias de este desarrollo tan rápido repercute en la generación de documentación sobre la aplicación. La ausencia de especificaciones precisas, motivan que posteriormente, haya que aplicar algún tipo de técnica para recuperar esa documentación abstracta que es tan necesaria a las tareas de mantenimiento de la aplicación.

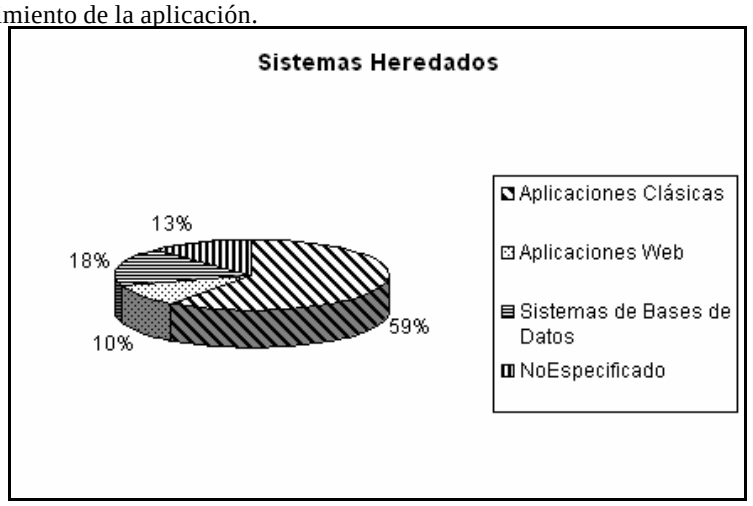

Fig. 2. Proporción de los tipos de sistemas heredados encontrados

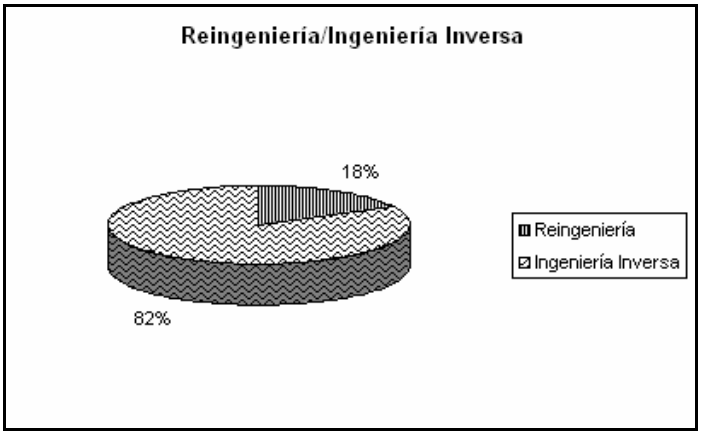

Fig. 3. Porcentaje de trabajos relacionados con la ingeniería inversa y la reingeniería

La Fig. 3 resume la proporción de trabajos dedicados a la ingeniería inversa y a la reingeniería del software. Como se puede observar a simple vista, la ingeniería inversa ocupa más del 80% del volumen total en cuestión de investigación y

publicaciones, dejando menos de un 20% de los trabajos para la reingeniería del software. Una de las razones por las que puede ocurrir esto, es que en todo proceso de reingeniería, siempre hay una etapa de ingeniería inversa, mientas que la ingeniería inversa puede llevarse a cabo de manera separada, sin que pretendamos emprender a posteriori una etapa de ingeniería directa.

Como ya se ha mencionado, la etapa de mantenimiento será más cara y costosa en función de la pobreza de la documentación de la que este dotado el software que se pretende mantener. A la vista de la Fig. 3, este hecho parece una buena razón para que los ingenieros realicen un esfuerzo extra en la recuperación de este tipo de especificación que bien por el paso del tiempo, bien por que no fueron desarrolladas, no están disponibles a la hora de mantener el sistema.

Las tareas relacionadas con la ingeniería inversa y por ende, con la reingeniería, suele ser complejas y pesadas debido a que en muchos casos, el substrato matemático en el que están basadas algunas de estas técnicas hace que su aplicación manual sea prácticamente inabordable. Por esto, y como vemos en la Fig. 4, son muchos los autores que proponen alguna herramienta que ofrezca soporte al método del que versa su publicación. El disponer de una herramienta o familia de herramientas que automatice, de forma total o parcial el proceso de recuperación de las especificaciones abstractas, revoca en un ahorro importante de tiempo y dinero, además de facilitarnos la comprensión del mismo mediante la posibilidad de visualizar el sistema mediante distintos puntos de vista.

El que haya un 38% de trabajos que no propone herramienta, se debe a que un porcentaje considerable de los mismos se haya en un estado preliminar, o sólo pretenden mostrar en su artículo el aspecto teórico de su propuesta. Por otro lado, el 62% de los trabajos incluyan una herramienta para llevar a cabo las tareas de reingeniería o ingeniería inversa, vienen a confirmar la importancia de automatizar estas tareas.

Fig. 4. Proporción de trabajos que proponen herramienta para la reingeniería y la ingeniería inversa

Entre la reingeniería y la ingeniería inversa, diríamos que la segunda se caracteriza por una mayor complejidad, ya que en muchos casos la recuperación de los modelos conceptuales del sistema conlleva el uso de complejos algoritmos que son, no obstante, necesarios. La complejidad de la segunda fase de la reingeniería, la

ingeniería directa, depende más de la labor que se quiera realizar a partir de las especificaciones obtenidas mediante la ingeniería inversa o de la metodología de trabajo que se vaya a utilizar. Al final, la automatización de las propuestas de los autores es casi tan importante como las propuestas en sí.

En la etapa de ingeniería inversa, la extracción de las especificaciones de mayor nivel de abstracción se realiza mediante análisis estáticos y dinámicos, aunque existen propuestas en las que se implementan los dos tipos. Mientras que el análisis estático consiste en la exploración del código fuente para extraer conocimiento en forma de especificaciones de más alto nivel, el análisis dinámico del código de una aplicación se basa en el seguimiento de la ejecución, generando las descripciones de alto nivel a partir de la información obtenida de la traza.

Algunas de las técnicas utilizadas con más frecuencia la ingeniería inversa, según el análisis realizado son:

- Grafos y patrones (Balanyi et al. 2003; Sartipi 2003):
- ASG (Abstract Syntax Graph).
- Patrones de búsqueda en grafos.
- Correspondencia de grafos.
- Algoritmos de clústering .
- Métodos matemáticos (Johnson et al. 2003; Villavicencio 2003).
- Interacción entre los usuarios y el sistema heredado (Stroulia et al. 2002).
- Parsers y Gramáticas (Draheim et al. 2003; Moonen 2003).
- Slicing Condicional (Villavicencio 2003).
- Técnicas y algoritmos de Clustering (Mitchell 2003a; Saeed et al. 2003).
- Modelos de reflexión jerárquica (Koschke et al. 2003).
- Reglas de transformación formales y metamodelos.(Briand et al. 2003).
- Métodos para el desensamblado de instrucciones (Schwarz et al. 2002).
- Técnicas basadas en contenedores (Wrapper) (Thiran et al. 2002).

5. CONCLUSIONES Y TRABAJO FUTURO

El mantenimiento es uno de los principales procesos del ciclo de vida del software, y como tal debe ser investigado en profundidad debido a su especial impacto en los sistemas de información de las organizaciones. De hecho, como señala (O'Neill 1997): "*La gestión eficiente del mantenimiento del software es una de las claves del éxito empresarial en la actualidad, ya que los sistemas de información y las tecnologías de la información son vitales para la salud de una organización. Las empresas que incrementen su habilidad para mejorar el mantenimiento de software como una competencia central mejorarán su posición competitiva*".

Dentro de las soluciones técnicas del mantenimiento del software destacan la ingeniería inversa y la reingeniería. En los últimos tres años se ha desarrollado una gran actividad investigadora en esta área, que hemos clasificado y resumido en este trabajo.

En general cabe destacar aspectos como que todavía son pocos los trabajos que abordan la reingeniería de sistemas heredados hacia aplicaciones Web basadas en componentes (García-Rodríguez de Guzmán et al. 2004) dedicándose todavía la gran

mayoría a aplicaciones comerciales que, habiéndose implementado mucho tiempo atrás siguen en uso actualmente, como son los tradicionales sistemas de información basados en COBOL.

También cabe destacar que, en un gran porcentaje de los estudios, se proponen (bien existentes en el mercado, bien de propio desarrollo) herramientas que ofrecen soporte a la propuesta teórica que exponen los autores en sus publicaciones.

Otro campo que quizá quede un poco desierto es la transformación de arquitecturas de los sistemas software, esto es, la migración de la arquitectura actual de un sistema de información hacia otras más adecuadas al momento tecnológico que se esté viviendo. El uso de metamodelos sería una potente herramienta para tal fin, a pesar de no haber sido utilizado con demasiada frecuencia entre los trabajos analizados.

Somos conscientes que los resultados aquí presentados deben ser ampliados con un estudio a mayor escala, que incluya otras conferencias de ingeniería del software (como, por ejemplo, ICSE) así como diversas publicaciones en revistas (p. ej. IEEE TSE o IEEE Software) en las que también pueden encontrarse trabajos relacionados con la ingeniería inversa y la reingeniería. Este estudio nos ayudará a comprender qué áreas se merecen una mayor atención por parte de los investigadores y a encontrar nuevas oportunidades para contribuir a la evolución de los sistemas software de las organizaciones.

6. REFERENCIAS

Alhajj, R. y F. Polat (2001). Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data. Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01), IEEE Computer Society,335-344.

Alvaro, A., D. Lucrédio y V. Cardoso García (2003). Orion-RE: A Coponent-Based Software Reengineering Environment. 10th Working Conference on Reverse Engineering (WCRE´03), IEEE Computer Society,248-259.

Arnold, R. S. (1992). Software Reengineering IEEE Press.

Balanyi, Z. y R. Ferenc (2003). Mining Desing Patterns from C++ Source Code. International Conference on Software Maintenance (ICSM'03), Amsterdam, The Netherlands, IEEE Computer Society,305-314.

Bennet, K. H. y V. T. Rajlich (2000). Software Maintenance and Evolution: a Roadmap. The Future of Software Engineering, International Conference on Software Engineering (ICSE'2000), Limerick (Irlanda), Finkelstein, A.,73-87.

Beyer, D., A. Noack y C. Lewerentz (2003). Simple and Efficient Relational Querying of Software Structures. 10th Proceedings Working Conference on Reverse Engineering (WCRE´03), IEEE Computer Society,216-227.

Bianchi, A., D. Caivano, V. Marengo y G. Visaggio (2001). Iterative Reengineering of Legacy Functions. IEEE International Conference on Software Maintenance (ICSM'01), Florence, Italy, IEEE Computer Society,632-641.

Blaha, M. (2001a). A Retrospective on Industrial Database Reverse Engineering Projects-Part 1. Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01), Suttgart, Germany, IEEE Computer Society,136-147.

Blaha, M. (2001b). A Retrospective on Industrial Database Reverse Engineering Projects-Part 2. Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01), Suttgart, Germany, IEEE Computer Society,147-156.

Briand, L. C., Y. Labiche y Y. Miao (2003). Towards the Reverse Engineering of UML Sequience Diagrams. Proceedings of the 10th Working Conference in Reverse Engineering, IEEE Computer Society

Bychkov, Y. y J. H. Jahnke (2001). Interactive Migration of Legacy Databases to Net-Centric Technologies. Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01), IEEE Computer Society,328-334.

Chikofsky, E. J. y J. H. Cross (1990). "Reverse Engineering and Desing Recovery: A Taxonomy." IEEE Software(January): 13-17.

Di Lucca, G. A., M. Di Penta, G. Antoniol y G. Casazza (2001a). An Approach for Reverse Engineering ofWeb-Based Applications. Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01), IEEE Computer Society,231-240.

Di Lucca, G. A., M. Di Penta, G. Antoniol y G. Casazza (2001b). An Approach for Reverse Engineering ofWeb-Based Applications. Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01), IEEE Computer Society

Di Lucca, G. A., A. R. Fasolino, F. Pace, P. Tramontana y U. De Carlini (2002). WARE: a tool for the Reverse Engineering of Web Applications. Sixth European Conference on Software Maintenance and Reengineering (CSMR´02), Budapest, Hungary, IEEE Computer Society,241-250.

Draheim, D., E. Fehr y G. Weber (2003). JSPick - A Server Pages Design Recovery Tool. Seventh European Conference on Software Maintenance and Reengineering (CSMR'03), Benevento, Italy, IEEE Computer Society,230-238.

Eisenbarth, T., R. Koschke y D. Simon (2001). Aiding Program Comprehension by Static and Dynamic Feature Analysis. IEEE International Conference on Software Maintenance (ICSM'01), Florence, Italy, IEEE Computer Society,602-611.

Ferenc, R., Á. Beszédes, M. Tarkiainen y T. Gyimóthy (2002). Columbus - Rerverse Engineering Tool and Schema for C++. International Conference on Software Maintenance (ICSM'02), Montreal, Quebec, Canada, IEEE Computer Society,172-181.

García-Rodríguez de Guzmán, I., M. Polo y M. Piattini (2004). Automated Generation of Component-Based Web Applications from Databases. Proceedings of the International Conference on Software Engineering and Practice, Las Vegas, Nevada, CSREA Press,372-377.

Henrad, J., J.-M. Hick, P. Thiran y J.-L. Hainaut (2002). Strategies for Data Reengineering Proceedings of the 9h Working Conference on Reverse Engineering, Richmond, Virginia, IEEE Computer Society

Henrard, J. y J.-L. Hainaut (2001). Data dependency elicitation in database reverse engineering Fifth European Conference on Software Maintenance and Reengineering (CSMR´01), Lisbon, Portugal, IEEE Computer Society,11-19.

Johnson, M. y C. N. G. Dampney (2003). Experience in developing interoperations among legacy information systems using partial reverse engineering International Conference on Software Maintenance (ICSM'03), Amsterdam, The Netherlands, IEEE Computer Society,369-372.

Knodel, J. y M. Pinzger (2003). Improving Fact Extraction of Framework-Based Software Systems. Proceedings of the 1oth Working Conference on Reverse Engineering, IEEE Computer Society

Koschke, R. y D. Simon (2003). Hierarchical Reflexion Models. 10th Working Conference on Reverse Engineering, Victoria, B.C., Canada, IEEE Computer Society

Lin, Y., R. C. Holt y A. J. Malton (2003). Completness of a Fact Extract. Proceedings of the 10th Working Conference on Reverse Engineering (WCRE´03), IEEE Computer Society,196-205.

Mancoridis, S., T. S. Souder, Y.-F. Chen, E. R. Gansner y J. L. Korn (2001). REportal: A Web-based Portal Site for Reverse Engineering. Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01), Suttgart, Germany, IEEE Computer Society,221-230.

Martin, J. y H. A. Müller (2001). Strategies for Migration from C to Java. Fifth European Conference on Software Maintenance and Reengineering (CSMR´01), Lisbon, Portugal, IEEE Computer Society,200-209.

Mitchell, B. S. (2003a). A Heuristic Approach to Solving the Software Clustering Problem. ICSM´03, Amsterdam, The Netherlands, IEEE Computer Society

Mitchell, B. S. (2003b). A Heuristic Approach to Solving the Software Clustering Problem. International Conference on Software Maintenance (ICSM'03), Amsterdam, The Netherlands, IEEE Computer Society,285-288.

Moonen, L. (2003). Exploring Software Systems. ICSM´03, Amsterdam, The Netherlands, IEEE Computer Society,276-280.

O'Neill, D. (1997). "Software Maintenance and Global Competitiveness." Journal of Software Maintenance: Research and Practice **9**(6): 379-399.

Piattini, M., F. Ruiz, M. Polo, J. Villalba, T. Bastanchury, M. A. Martínez y C. Nistal (2000). Mantenimiento del Software: Modelos, Técnicas y Métodos para la Gestión del Cambio. Spain, RA-MA.

Pigoski, T. M. (1997). Practical Software Maintenance. Best Practices for Managing your Investment. Estados Unidos, John Wiley & Sons.

Pinzger, M., M. Fischer, H. Gall y M. Jazayeri (2002). Revealer: A Lexical Pattern Matcher for Architecture Recovery. Proceedings of the 9th Working Conference on Reverse Engineering, Richmond, Virginia, IEEE Computer Society

Riva, C. y J. Vidal Rodriguez (2002a). Combinig Static and Dynamic Views for Architecture Reconstruction. Sixth European Conference on Software Maintenance and Reengineering (CSMR´02), Budapest, Hungary, IEEE Computer Society,47-58.

Riva, C. y Y. Yang (2002b). Generation of Architectural Documentation using XML. Proceedings of the 9th Working Conference on Reverse Engineering (WCRE´02), Richmond, Virginia, IEEE Computer Society,161-169.

Riva, C. y Y. Yang (2002c). Generation of Architetural Documentation using XML. Proceedings of the Ninth Conference on Reverse Engineering (WCRE´02), IEEE Computer Society

Saeed, M., O. Maqbool, H. A. Babri, S. Z. Hassan y S. M. Sarwar (2003). Software Clustering Techniques and the Use of Combined Algorithm. Seventh European Conference on Software Maintenance and Reengineering (CSMR'03), Benevento, Italy,301-306.

Sartipi, K. (2003). Software Architecture Recovery based on Pattern Matching. ICSM´03, Amsterdam, The Netherlands, IEEE Computer Society,293-297.

Sartipi, K. y K. Kontogiannis (2001). A Graph Pattern Matching Approach to Software Architecture Recovery. IEEE International Conference on Software Maintenance (ICSM'01), Florence, Italy,408-421.

Sartipi, K. y K. Kontogiannis (2003). On Modeling Software Architecture Recovery as Graph Matching. International Conference on Software Maintenance (ICSM'03), Amsterdam, The Netherlands, IEEE Computer Society,224-234.

Schwarz, B., S Debray y G. Andrews (2002). Disassembly of Executable Code Revisited. Proceedings of the 9th Working Conference on Reverse Engineering (WCRE´02), Richmond, Virginia, IEEE Computer Society,45-54.

Stoermer, C., L. O´Brien y C. Verhoef (2002). Practice Patterns for Architecture Reconstruction. Proceedings of the 9th Working Conference on Reverse Engineering (WCRE´02), Richmond, Virginia, IEEE Computer Society,151-160.

Stroulia, E., M. El-Ramly y P. Sorenson (2002). From Legacy to Web through Interaction Modeling. International Conference on Software Maintenance (ICSM'02), Montreal, Quebec, Canada, IEEE Computer Society,320-331.

Thiran y J.-L. Hainaut (2001). Wrapper Development for Legacy Data Reuse. Eighth Working Conference on Reverse Engineering (WCRE'01), Suttgart, Germany, IEEE Computer Society,198-207.

Thiran y J.-L. Hainaut (2002). Wrapper Development for Legacy Data Reuse. Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01), Suttgart, Germany, IEEE Computer Society

Tonella, P. y A. Potrich (2001). Reverse Engineering of the UML Class Diagram from C++ Code in Presence of Weackly Typed Containers. IEEE International Conference on Software Maintenance (ICSM'01), Florence, Italy, IEEE Computer Society,376-385.

Tonella, P. y A. Potrich (2002). Static and Dynamic C++ Code for the Recovery of the Object Diagram. International Conference on Software Maintenance (ICSM'02), Montreal, Quebec, Canada, IEEE Computer Society,54-65.

Vanderdonckt, J., L. Bouillon y N. Souchon (2001). Flexible Reverse Engineering of Web Pages with VAQUISTA. Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01), IEEE Computer Society,241-248.

Villavicencio, G. (2003). Formal Program Reversing by Conditioned Slicing Seventh European Conference on Software Maintenance and Reengineering (CSMR'03), IEEE Computer Society,368-378.

Wilde, N., H. Page y V. Rajlich (2001). A Case Study of Feature Location in Unstructured Legacy Fortran Code. Fifth European Conference on Software Maintenance and Reengineering (CSMR´01), Lisbon, Portugal, IEEE Computer Society,68-76.

Yeh, D. y W.-Y. Kuo (2002). Reverse Engineering Aggregation Relationship Based on Propagation of Operations. Sixth European Conference on Software Maintenance and Reengineering (CSMR´02), Budapest, Hungary,223-229.