Reverse Engineering of Web Applications: A Technical Review

Reshma Patel[1]
Frans Coenen[1]
Russell Martin[1]
Lawson Archer[2]

[1] The University of Liverpool, Department of Computer Science, Ashton street, Liverpool L69 3BX
{reshma, frans, martin}
@csc.liv.ac.uk

[2]Transglobal Express Ltd, Unit 5, The Gateway, Wirral International Business Park,Bromborough, Wirral, CH62 3NX
Lawson@
transglobalexpress.co.uk

June / July 2007

REVERSE ENGINEEING OF WEB APPLICATIONS: A TECHNICAL REVIEW

Reshma Patel[1], *Frans Coenen*[1], *Russell Martin*[1], *Lawson Archer*[2]
[1]Department of Computer Science, The University of Liverpool, Ashton St. Liverpool L69 3BX
{reshma, frans, martin} @csc.liv.ac.uk
[2]Transglobal Express Ltd, Unit 5, The Gateway, Wirral International Business Park, Bromborough, Wirral, CH62 3NX
lawson@transglobalexpress.co.uk

Abstract

The World Wide Web (WWW) has become one of the most important means of communication for commercial enterprises of all kinds. This is particularly the case in commercial sectors where on-line sales often represent a significant proportion of the total sales. The current sophistication of World Wide Web services is such that some businesses have elected to conduct all their sales on-line. Regardless of the individual degree of commitment to on-line commerce most commercial enterprises acknowledge that it is important to have at least a WWW presence. In the rush to obtain such an on-line presence, and subsequently maintain and expand this presence, well established software engineering practices applied to the production of more traditional software system have often been overlooked. This can partly be attributed to difficulties in adapting traditional software engineering practices to WWW software, and partly to the "rush to market" commercial pressures. The result has been that many WWW applications are poorly documented (or not documented at all) and poorly structured, which in turn has made it very difficult to maintain these system. Much work has been done to establish sound software engineering techniques for the construction of WWW systems. However, with respect to the many systems built without the benefit of such software engineering practices, programmers charged with maintenance of these systems have been forced to first reverse engineer these systems. Reverse engineering is a time consuming and mundane task that entails none of the "glamour" associated with the construction of software systems. In the case of WWW applications the "state-of-the-art" is still in its infancy. This paper presents a technical review of the current "state of the art" of WWW Application (WA) revere engineering.

1. Introduction

The growth of the Internet over the last decade has caused it to develop into a platform where differing applications are able to handle complex transactions, thereby providing a ubiquitous environment where user experience is improved to a significant level.

Web Applications (WAs) have become popular with respect to both, large and small-medium enterprises (SMEs) as they provide the underlying engines that not only improve a company's image, but also act as a useful resources for increasing a company's overall market share. Such systems therefore act as valuable assets for companies and organisations, and should not be spontaneously put together, despite tight development schedules. However, the prerequisite for complying with a sound software development lifecycle is not considered in most cases of WA development. Di Luuca notes that the *"analysis and design documentation in existing WAs is very poor, if not completely absent"* due to the *"pressure of short time to market and extremely high competition"* (Lucca et al 2006). Pessman (writing in 2000) makes a similar observation, noting that the reluctance of WA developers to adopt well proven software engineering principals is worrisome. As consequence many WAs are expected to be short lived as the lack of appropriate documentation and engineering makes them harder and harder to maintain. There are a number of reasons for this, but the most significant are:

1. "Rush to market" commercial pressure as a result of which developers are not afforded adequate development time.

2. The rate of change of the technology which has meant that the availability of appropriate development tools has not always kept pace with the continuously advancing technological changes, in other words, appropriate development tools and techniques are not always available.

Although WAs are dominant, they are not the only applications that have been developed without the benefit of sound software engineering practices. Many organsistions still operate *legacy* systems developed before the significance of adopting sound software engineering strategies was fully realised. The discipline of reverse engineering was first proposed to address the "inherited" problems associated with legacy software so as to provide users of such systems with the option of updating and extending such software systems rather than allowing them to become redundant.

Although both legacy systems and WAs suffer from poor structure and lack of documentation, the distinction between the reverse engineering of these systems is in the objective of the reverse engineering. In the case of a legacy system the poor structure and lack of documentation is a function of age; as the system has been adapted and extended over time additional code modules have been continuously added, without the benefit of proper engineering, with the objective providing "quick fixes". The aim of reverse engineering in this case is typically to provide a foundation to support the reengineering of the system (for example because it is no longer compatible with modern systems with which it needs to interface, e.g. databases, internet, etc). In the case of WAs the poor structure and lack of documentation is a function of commercial pressures to "get the system up and running". In this second case (our case) we wish to reverse engineer the system so that we can discover how it works so that we can maintain (evolve) it, and we do not wish to reimplement it as it is already implemented using current technology and (typically) is comparatively recent. Note that in this context Boldyreff and Kewish (2001) identify a set of potential WA maintenance tasks.

The reverse engineering of "traditional" legacy systems is not the principal concern of this paper, because (i) the reverse engineering of such systems is reasonably well understood, and (ii) with the further passage of time it can be expected that there will be fewer and fewer of these systems. The focus of this paper is the reverse engineering of WAs, an area that has received much less attention because it was not perceived as an issue until more recently. The earliest reference to WA reverse engineering is arguably Antoniol et al. 2000, more traditional reverse engineering has been around since the late 1980s, for example , Sneed (1984) provides an early reference.

WA reverse engineering should also not be confused with design reuse (see for example Schwabe et al., 2001), the process of using a basic foundation design and using this to engineer many WAs. Finally some work has also been done on Web-based (or on-line) reverse engineering systems which are WAs designed to support traditional reverse engineer. One example of Web-based reverse engineering system is REportal (Mancoridis et al., 2001).

The paper is structured as follows: In Section 2 the authors present a general overview of the discipline of reverse engineering. Section 3 provides a brief categorisation of WA development methodologies (on which many WA reverse engineering ideas are based). A categorisation of WAs is presented in Section 4 together with a review of the objectives of WA reverse engineering in Section 5. Some brief consideration is given in section 6 to the representation of the architecture/structure of WAs. In Section 7 the authors report on the specific tools and methodologies that can be used for extracting conceptual and technical information from WAs to aid the reverse engineering process. Finally some conclusions are presented in Section 8.

2. Reverse Engineering

Chikofsky and Cross (1990) describe reverse engineering as *"the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction"*. Their pioneering work on this subject area suggests that the main reason that reverse engineering is frequently adopted is because it enables engineers to recover inadequate or non-existent information about a system by conducting a "backward" analysis of the existing legacy system in order to gain an overall perspective of its functionality. This in turn, facilitates the production of accurate documentation describing the original implementation, design and requirements stages of the lifecycle. For further information on "standard" reverse engineering the authors would recommend Muller et al. (2000) who, although now somewhat dated, present a good review of reverse engineering activities. A more recent, high level, overview of the aims and objectives of traditional reverse engineering and the associated issues can be found in Deursen and Burd (2005). An interesting discussion of early work on traditional reverse engineering approaches can be found in Nelso (1996).

Many WWW reverse engineering methodologies/tools are built on established (general) reverse engineering approaches. For example Benedusi et al. (1992) describe a (general) reverse engineering paradigm called GMT (Goals, Models, Tools) which was later used to support the WARE (Di Lucca et al. 2004) WA reverse engineering methodology (see below). Hassan and Holt (2001) describe how they adapted the Portable Bookshelf Environment (PBS) for "stand alone" reverse engineering to derive the architecture of WAs. Boldyreff and Kewish (2001) propose a system that exploits traditional reverse engineering ideas to extract duplicate content and style from WWW pages with the aim of restructuring the WWW pages so as to improve their maintainability.

Even though the concept of reverse engineering has become widely accepted, this subject area in general has also brought about some controversy in terms of legal concerns regarding disassembling of a third parties' software application, especially in the context of copyright and intellectual property rights (Samuelson, 2002). However, exclusive rights for organisational applications developed by third-party developers are, in most cases, held by the organisations themselves and they therefore act as the owner of the software. In such cases, Kienle et. al. (2004) determined that any manipulative activities performed on the system are not regarded as illegal acts as long as ownership rights and regulations are established at an early stage.

3. WA Development Methodologies

Many WWW reverse engineering methodologies are built on WWW application development methodologies (which in turn are built on existing general software development methodologies and tools sets such as UML). Some examples of such WA development methodologies include:

1. There are a number UML based methodologies. For example Conallen (1999) who proposed (amongst other intuitions) a number of extensions to UML class diagrams to aid the development of WWW applications. Some of the ideas proposed by Conallen were subsequently included in the WARE system for the reverse engineering of WWW applications (Di Lucca et al. 2002a and 2004) which is further described in Section 7.
2. The Relationship Management Methodology (RMM) of Isakowitz et al. (1997) derived for WA applications, particularly those that interface with relational database. The central feature of RMM is the Relationship Management Data Model (RMDM) which provides for an abstract description of a WA. Antoniol et al. (2000) describe a WA reverse reengineering approach founded on RMM (this is described in more detail in Section 7 below.

4. Classification of Web Applications

To provide an insight into the kind of information and related associations that need to be captured during a WA reverse engineering process, it is important to understand the types of web applications that exist. Tilley and Huang (2001) provide a useful categorisation of WAs into three classes:

1. **Class 1:** Static applications, typically implemented in HTML with no user interaction
2. **Class 2:** Client side interaction with Dynamic HTML (DHTML), typically using mouse clicks
3. **Class 3:** Contain dynamic content created "on-the-fly". Typically use technologies such as JSP, Java Servelets, PHP, ASP, ODBC, JDBC etc.

The above classes thus categorise the complexity of *Client* and/or *Server* pages and can be used to form the basis for describing the level of interaction and dynamicity provided within a WA. Lucca (2004) concisely defines a client page as one which is sent to a client as a result of a request and can be further classified as being either a) *static* or b) *client built*.

The degree of complexity and complication associated with a reverse engineering process tends to increase when reverse engineering the higher of the above categories (classes 1 and 2) where a user's interaction and manipulation activities across a system are more frequent. Static applications tend to be non-chaotic and therefore can be easily modelled. Most commercial WAs are not static as it is seen as desirable to provide customer service via a dynamic application so as to make customers feel "in control" and "part of" the organisation (Lucca et. al., 2004). Dynamic customer services are considered to enhance user satisfaction (compared to static services) with consequent commercial benefits for the company.

For this reason the reverse engineering of Class 3 WAs requires considerable more effort than that associated with Class 1 and 2 WAs. Tools that support (i) the systematic abstraction and analysis of components, (ii) the identification of relations and (iii) architectural modelling, can thus provide considerable benefits with respect to Class 3 WAs. The nature of such tools will be discussed further in the following Section.

5. WA Reverse Engineering Objectives

Chikofsky and Cross (1990), referring to what we in this paper call traditional or standard reverse engineering, state that "The primary purpose of reverse engineering a software system is to increase the overall comprehensibility of the system for both maintenance and new development".

In the context of the objectives of WA reverse engineering, the work described in Tiley (1998) is of interest. Tilley describes a number of high level aspects and requirements for reverse engineering environments. Tilley starts by considering the cognitive processes required to form a mental representation of the software system under consideration. In this context he identifies three approaches: (i) bottom up, (ii) top down and (iii) opportunistic (a combination of the first two); Tilley suggests that the latter is the most productive. He then goes on to identify a number of traditional reverse engineering objectives:

1. **Pattern Abstraction**: Three levels:
   a. Programme Analysis (the most common form of reverse engineering).
   b. Plan Recognition, aimed at identifying instances of commonly used structures, and resulting in an abstract representations of fragments of source code.
   c. Concept Assignment, the process of discovering human inspired concepts and linking then to implementational concepts.

2. **Redocumentation**: The process of generating accurate documentation from existing, undocumented software. Essentially a transformation from source code to Pseudo code, and arguably the oldest form of reverse engineering (cf. Sneed 1984).
3. **Architecture Recovery**: Obtaining an understanding of the structural aspects of a system's architecture.

Tilley characterises the reverse engineering processes that we might wish to undertake under three headings:

1. **Data**. Data gathering is usually the first RE process. Techniques used fore data gathering include (i) system examination, (ii) document scanning and (iii) experience capture. System examination usually comprises static analysis of the source code (the alternative is dynamic analysis which requires the running/execution of the code).
2. **Knowledge**. Knowledge management (a popular research area in the late 1990s) is concerned with the capture and organisation of knowledge about the problem domain.
3. **Information**. Information exploration where the majority of programme understanding takes place as part of information exploration.

Tilley ends by identifying a number of desirable quality attributes for reverse engineering environments which are later used (Tilley and Huang, 2001) to evaluate a number of WA reverse engineering environments (see below).

Generally speaking WA reverse engineering activities tend to concentrate on the static analysis of source data rather than the dynamic analysis of the operation of WAs. Schwabe et al. (2001) identify three areas of concern when undertaking WA (forward) engineering which are equally applicable to WA reverse engineering:

1. Application behaviour.
2. Navigational modelling.
3. Interface design.

6. WA Representation

To facilitate reverse engineering Was, use is typically made of a number of kinds of data structure. Two popular representations are tabular and graph representations. Ricca and Tonella (2001) propose a graph model for representing WA architectures such that the WA is represented by a directed graph `W=(P,E)` where `P` is a set of HTML pages and `E` a set of links connecting members of `P`. Ricca and Tonella then apply flow analysis to the graph structure to identify various features.

7. Web Application Reverse Engineering Tools and Methodologies

Like software engineering where ad-hoc development is not acceptable and professional methods must be utilised to solve a problem and increase project success rate, it is important that reverse engineers of WAs follow the same disciplined structure. The use of dedicated WA reverse engineering methodologies should provide a consistent and complete plan, making it possible to determine which processes need to be carried out together with associated tools required for supporting the methods.

For simple/small WAs manual analysis is sufficient. However, in the case of large WAs, which may comprise several thousand lines of code (LOC) and various interlinked modules, it is not practical to manually analyse the system. Instead computer-aided tools will be required to

minimize costs while at the same time ensuring that maximum information is extracted for model creation and documentation of the functionality, behaviour, dependencies and structure of the system.

In the literature a number of dedicated WA reverse engineering methodologies and tools are described. Those that the authors are aware of are briefly reviewed in the following subsections. The authors have grouped the various reported systems under four headings:

1. UML Based Methodologies
2. Specific ASP.NET Methodologies
3. Ontology Based WA reverse engineering.
4. Other WWW application RE Methodologies/Tools.

There are very few published reviews of WA reverse engineering. However, Tilley and Huang (2001) present an evaluation of three WWW development tools (Microsoft FrontPage 2000, Macromedia Dreamweaver UltraDev 4 and Big Picture Technologies SmartSite 3) with respect to the support they provide for a number of WWW RE activities including:

1. Redocumentation.
2. Data gathering (identification of the artefacts and relationships of a WWW system).
3. Knowledge management (capturing the knowledge used to build a WWW system).
4. Information exploration (ability to gain an in depth understanding).

The conclusions drawn are related to the requirements for future WA reverse engineering tools rather than the capabilities of the studied systems.

7.1. UML-Based Methodologies

According to the research carried out by the authors, it was found that the majority of reported WA reverse engineering methodologies and tools are founded on the Unified Modelling Language (UML), which is the most widely accepted modelling standard adopted during the forward engineering design process (for both traditional and WA applications). UML-based technique provides a stable, familiar environment for reengineers to work with for modelling components as well as the *behaviour* of an application, thus shortening the learning curve involved when integrating such methods/tools during the analysis process. Examples of UML-based methods can be found in the work of a diverse set of researchers such as Chung and Lee (2000), Pu et. al. (2003) and Di Lucca et. al. (2004).

A very well documented example is the WARE (Web Application Reverse Engineering) tool developed by Di Lucca (Di Lucca et al. 2004, Di Lucca et al. 2001). This methodology is based on Benedusi's "Goals, Models and Tools" (GMT) paradigm (Benedusi et. al., 1992) and adopts the UML extensions proposed by Conallen (1999) to extract package diagrams for detailing dynamic information. The modelling phase includes the use of: (i) UML use-case diagrams to specify functional requirements, (ii) UML class diagrams to describe the structure of WAs and (iii) UML sequence diagrams to document the dynamic interaction. Additionally, Di Lucca et. al. also devised an associated WARE tool, simply known as the 'WARE-tool' (Di Lucca et al. 2002a), to provide support for this methodology, thus offering an extensive package for reverse engineering web systems. An analysis of WARE is presented in Di Lucca et al. (2002b). The WARE project was eventually written up, by Porfirio Tramontana, into a PhD thesis which is summarised in Tramontana (2005).

Cung and Lee (2000(also adopt the Conallen extensions (Conallen, 1999) and extract component diagrams (each WWW page is a component) and package diagrams (reflecting the WA directory structure).

The UML based methodology proposed by Pu et al. (2003) obtains domain knowledge from: (i) software parameters (names, types, lengths, etc), (ii) system calls (procedures and functions), (iv) operations and (v) notes (in line comments etc.). The acquired information is then mapped into UML class and use case diagrams.

7.2. Specific ASP.NET Methodologies

Many WAs are currently developed using Microsoft's ASP.Net technology. A number of reported WA reverse engineering systems are specifically directed at ASP.Net applications. Of note is the work of Katsimpa et al. (2006).

Katsimpa et al. describe a WWW application reverse engineering methodology specifically intended for applications created using ASP.NET; although, as the authors point out, it also has general applicability. ASP WWW applications typically comprise of a collection of ASP.NET pages and forms (the latter stored in `aspx` files) supported by further code, configuration files and XML meta-data description files. Static analysis is undertaken using two parsers. The first is applied to the `aspx` files and produces a tree for each page. The process is as follows:

1. Pre-process `aspx` files to ensure that they are well structured (i.e. start tags matched with end tags where appropriate, no cross nesting of tag pairs, inclusion of quotes for tag parameters etc.).
2. Prepare a table/matrix of nodes of interest.
3. For each well structured `aspx` page create an ASP.NET tag trees.
4. Match labels in table with nodes in the trees, for each matched node update entry in text file (one text file per tree). Each entry includes as appropriate type unit corresponding to the tag and the object ID that represents the node. Each object in an `aspx` page has a unique ID. The result is a mapping of HTML and ASP controls into WebML units.
5. Process text file and analyse corresponding source code. Aim is to identify SQL keywords (SELECT, UPDATE, etc.) consequently specific database tables.
6. Traverse the application directories and represent this as another tree. Leaf nodes in this tree contain information from the previous step.
7. Add further edges to the final tree representing links between units and pages ore units and units.

This approach is also supported by a tool developed by Katsimpa, which, just like the WARE tool, provides a complete package for stability and uniformity.

7.3. Ontology Based WA reverse engineerig

The ontology approach to reverse engineering web systems has increased its popularity in the past few years, mainly because it *"provides a common referenceable set of concepts for use in communication"* (van Rees, 2003). The focal point of this method is that it models WAs via a schema. Benslimane et. al (2006, see also Bouchiha et al 2007) propose an approach known as 'OntoWer', The objective of OntoWare is to enable conceptual schemas to be created. The foundation of the work of Benslimane et al. is the view that current research contributions on reverse engineering do not provide adequate knowledge and support as they mostly tend to *"focus their discussion to a particular analysis technique, without clarifying its relationship to the problem of reverse engineering in general"* (a view also supported by Du Bois, 2006). The ontological approach caters for high-level analysis of a WA.

7.4. Other WWW application RE Methodologies/Tools

Hassan and Holt (2001) describe a WA reverse engineering founded on the Portable Book Shelf (PBS) environment. The start point for their approach is a set of documents, typically source code

of various kinds but also other types of document.. These are then processed using various *extractor* tools that are designed to "extract" facts about the given WA. Hassan and Holt mention at least five different kinds of extractor: (i) html, (ii) Server script, (iii) DB access, (iv) Language an (v) binary extractors. The generated facts are stored in a diagrammatic structure. The complexity of the generated diagrams is reduced by clustering facts, partly achieved automatically and partly by user intervention. The decomposed system is then presented using further diagrams.

Other documented WWW application reverse engineering methodologies/Tools include:

1. **ReWeb**: (Ricca and Tonella, 2000, 2001). A tool to undertake traditional source code analysis of WAs by representing the WA as a graph structure and undertaking various types of analysis such as reachability, flow and traversal analysis. ReWeb can download and analyse WAs. Various search and navigation tools are included and there is also am option to illustrate the evolution of WA using a colour coding. Popup windows display the outcome of the analysis.
2. **VAQUISTA** (Vanderdonckt et al. 2001, Bouillon and Vanderdonckt 2002) is directed at reverse engineering the user interface of WAs to facilitate their migration to different platforms. The VAQUISTA process commences with the static analysis of an HTML page and translating this into a *presentation model* describing the elements of the HTML page. The presentation model comprises a hierarchy of *presentation elements* or *objects*. Vanderdonckt et al. identify four categories of presentation object:
   a. Concrete Interaction Objects (CIOs), real elements in the HTML page that can not be decomposed further such as text, images, animations, push buttons, check boxes, etc.
   b. Abstract Interaction Objects (AIOs), abstractions of CIOs identified by a name, attributes and other orameters.
   c. Logical Window (LW) the logical container for AIOs or a physical window (dialog box, check box, etc).
   d. Presentation Unit (PU), the presentation environment for undertaking an interactive task.

   Transformations are undertaken using *mapping tables* that can be used in a variety of ways as directed by the user. (See also Vanderdonckt and Bouilllon, 2002).
3. **TERESA** (Paganelli and Paterno, 2002) is a source code statistical analysis tool that produces a task-oriented model of a Web application.
4. **Revangie** (Draheim et al.2005). performs "source code" independent reverse engineering of WAs, i.e. a *black box* approach to WA reverse engineering (as opposed to while box). Revangie uses a form-oriented user interface model that uses "typed bipartite state machines" to model (i) client pages and (ii) server actions. The models are graphs that include the relationships between server-side actions and pages. Revangie has three modes of operation depending on the degree of user interaction:
   a. **The crawl mode** which operates on the client side.
   b. **The snoop mode** which operates at points between the communication line between server and client.
   c. **The guide mode** , a combination of the crawl and snoop modes.
5. The **RMM** based methodology described in Antoniol et al. (2000) is a reengineering methodology, however the process commences with a reverse engineering stage. Antoniol et al. describe this process using a university module as an example where the WA is frame based. The process commences with the identification of links which are then used to build a Relationship Management Data Model (RMDM). From the RMDM, and further analysis, an Entity-Relationship (ER) model is abstracted. This is the end point for the reverse engineering.
6. **JSPick** (Draheim et al. 2003) is a reverse engineering tool directed at JSP (Java Server Pages), The tool is used to extract "page signatures and form types".

Hassan and Holt (2002, 2003) describe a set of extraction tools to analyse WA source code to produce "box-and-arrow" diagrams describing the WAs architecture. The aim is to produce a visualisation of a WA so as to facilitate better understanding and consequently maintenance.

Finally Girardi et al. (2002) describe techniques and algorithms to support the restructuring of multi-lingual WWW applications with a view to future maintenance. In Tonella et al. (2002) a similar approach founded on ReWeb (see above) is also used to support the restructuring of multi-lingual WAs so as to ensure consistency.

8. Conclusions

In this paper a review has been presented of the "state of the art" of WA reverse engineering practices, tools and techniques. The reversed engineering of WAs requires different procedures than that required for other software systems. This is because WAs are internet based and therefore are developed using a specific set of technologies not found in "stand alone" software systems. Another distinction is that the objective of reverse engineering as applied to other applications tends to be in the context of legacy system where we wish to reengineer the system. This is usually not the case with WAs which tend to be more recent and therefore fully operational, the aim here is to find out how it works (with a view to future maintenance) because the WA was not properly documented when it was first developed.

Although a substantial amount of work has been undertaken and reported in the literature on "traditional" reverse engineering, little work has been done on the reverse engineering of WAs. From the investigation reported here, the main findings are:

1. WA reverse engineering methodologies tend to be founded on either (i) more standard reverse engineering methodologies or (ii) recently developed WA development methodologies (which are themselves founded on standard software engineering practices).
2. The most common formalism is based on UML.

What is also clear is that the field of WA reverse engineering is still in its infancy with little agreement on standardised approaches or methodologies.

References

1. Antoniol, G., Canfora, G., Casazza, G. and De Lucia, A. (2000). *Web Site Reengineering Using RMM*. Proc. 2nd Int. Workshop on Web Site Evolution, pp9-16.
2. Benedusi, P., Cimitile, A., De Carlini, U. (1992). *Reverse Engineering Process, Document Production and Structure Charts*. Journal of Systems and Software, Vol. 19, pp225-245.
3. Benslimane, M. S., Malki, M., Bouchiha, D., Benslimane, D. (2006). *OntoWer: An Ontology Based Web Application Reverse Engineering Approach*. International Review on Computers and Software, 1(1), pp52-58.
4. Boldyreff, C. and Kewish, R. (2001). *Reverse Engineering to Achieve Maintainable WWW Sites*. Proc. 8th Working Conference on Reverse Engineering, WCRE'01, IEEE, pp249-257.
5. Bouchiha, D., Malki, M. and Benslimane, S.M. (2007). *Ontology based Web Application Reverse-Engineering Approach*. INFOCOMP journal of Computer Science, 6(1), pp37-46.
6. Bouillon, L. and Vanderdonckt, J. (2002). *Retargeting of Web Pages to Other Computing Platforms with VAQUISTA*. Proc 9th Working Conference on Reverse Engineering, WCRE'02, IEEE, pp339-348.
7. Chikofsky, E. J., Cross, J. H. (1990) *Reverse Engineering and Design Recovery: A Taxonomy*. IEEE Software, Vol. 7 (10), pp13-17.
8. Chung, S., Lee, Y. S. (2001) *Reverse Software Engineering with UML for Web Site Maintenance*. Proceedings 1st Conference on Web Information Systems Engineering, IEEE, pp156-161.
9. Conallen, J. (1999). *Building Web Applications with UML*. Addison Wesley. ISBN: 0-201-61577-0.

10. van Deursen, A. and Burd, L. (2005). *Software Reverse Engineering*. Guest Editorial, Jo of System and Software, Vol 77(3), Special issue on Reverse Engineering, pp209-212.
11. Di Lucca, G.A., Di Penta, M., Antonniol, G. and Casazza, G. (2001). *An Approach for Reverse Engineering of Web-Based Applications*. Proc. 8th Working Conference on Reverse Engineering, WCRE'01, IEEE, pp231-240.
12. Di Lucca, G.A., Fasolino, A.R., Pace F., Tramontana, P. and De Carlini, U. (2002a). *WARE: A Tool for The Reverse Engineering of Web Applications*. Proc. 6th European Conference on Software Maintenance and Rengineering (CSMR'02), pp241-250.
13. Di Lucca, G.A., Fasolino, A.R. and Tramontana, P. (2002b). *Towards a Better Comprehensibility of Web Appl,ications: Lessons Learned from Reverse Engineering Experiments*. Proc. 4th Int Workshop on Web Site Evolution (WSE'02), IEEE, pp33-42.
14. Di Lucca, G.A., Fasolino, A.R. and Tramontana, P. (2004). *Reverse Engineering Web Applications: Thw WARE Approach*. Journal of Software Maintenance and Evolution, Research and Practice, Vol 16, pp71-101.
15. Draheim, D., Fehr, E. and Weber, G. (2003). JSPick – A Server Pages Design Recovery. Proc 7th IEEE European Conference on Software Maintenance and Reengineering, LNCS, pp230-236.
16. Draheim, D., Lutteroth, C. and Weber, G. (2005). A Source Code Independent Reverse Engineering Tool for Dynamic Web Sites. Proc. 9th European Conference on Software Maintenance and Reengineering (CSMR'05), pp168-177.
17. Du Bois, B. (2006). *Towards a Reverse Engineering Ontology*. 2nd International Workshop on Empirical Studies in Reverse Engineering (WESRE 2006), collocated at WCRE 2006.
18. Girardi, C., Pianta, E., Ricca, F. and Tonella, P. (2002). *Restructuring Multilingual Web Sites*. Proc. 18th Int. Conf. on Software Maintenance (ICSM'02), IEEE, pp290-299.
19. Hassan, A.E. and Holt, R.C. (2001). *Towards a Better Understanding of WWW Applications*. Proc. 3rd Int. Workshop on Web Site Evolution (WSE'01), IEEE, pp112-116.
20. Hassan, A.E. and Holt, R.C. (2002). *Architecture Recovery of Web Applications*. Proc. Int. Conf. on Software Engineering (ICSE'02), pp349-359.
21. Hassan, A.E. and Holt, R.C. (2003). *A Visual Architectural Approach to Maintaining Web Applications*. Anals of Software, Vol 16.
22. Isakowitz, T., Kamis, A. and Koufaris, M. (1997). *Extending the Capabilities of RMM: Russian Dolls and Hypertext*. Proc. 30th Hawaii Int. Conf. on System Science, pp166-186.
23. Katsimpa, T., Panagis, Y., Sakkopoulos, E., Tzimas, G., Tsakalidis, A. (2006) *Application Modelling using Reverse Engineering Techniques*. Proceedings Symposium on Applied Computing (SAC'06), pp1250-1255.
24. Kienle, M. H., German, D., Muller, H. (2004). *Legal Concerns of Web Site Reverse Engineering*. Sixth IEEE International Workshop on Web Site Evolution (WSE'04), pp41-50.
25. Mancoridis, S., Souder, T.S., Chen, Y-F., Gansner, E.R. and Korn, J.L. (2001). *REportal: A Web-based Portal Site for Reverse Engineering*. Proc 8th Wprking Comference on Reverse Engineering,, IEEE press, pp 221-230.
26. Nelson, L. M. (1996) *A Survey of Reverse Engineering and Program Comprehension*. ODU CS 551 – Software Engineering Survey.
27. Muller, H.A., Jahnke, J.H., Smith , D.B., Storey, M.A., Tilley, S.R. and Wong, K.(2000). *Reverse Engineering: A Roadmap*. Proc. ACM ICSE Future of Software Engineering Track, pp47-60.
28. Paganelli, L. and Paterno, F. (2002). *Automatic Reconstruction of the Underlying Interaction Design of Web Applications*. Proc. 14th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'02), ACM, pp439-445.
29. Pressman, R.S. (2000). *What a Tangled Web We Weave*. IEEE Software, 17(1), pp18-21.
30. Pu, J., Millham, R., Yang, H. (2003). *Acquiring Domain Knowledge in Reverse Engineering Legacy Code into UML*. Proceedings 7th IASTED International Conference on Software Engineering and Applications (SEA), ACTA Press.
31. van Ress. R. (2003) *Clarity in the Usage of the terms Classification, Taxonomy and Ontology*. Proceedings of the 20th International Conference on Information Technology for Construction.

32. Ricca, F. and Tonella, P. (2000). *Web Site Analysis: Structure and Evolution*. Proc. 16th IEEE Int. Conf. on Software Maintenance (ICSM'00), pp76-86.
33. Ricca, F. and Tonella, P. (2001). *Understanding and Restructuring Web Sites with ReWeb.* IEEE Multimedia, pp76-86.
34. Samuelson, P. (2002). *Reverse Engineering under Siege.* Communications of the ACM, Vol. 45, Issue 10, pp15-20.
35. Schwabe, D., Rossi, G., Esmeraldo, L. and Lyardet, F., (2001). *Engineering Web Applications for Reuse.* IEEE Multimedia, 8(1), pp20-31.
36. Sneed, H. (1984). *Software Renewal: A Case Study.* IEEE Software, 1(3), pp56-63.
37. Tilley, S. (1998). *A Reverse Engineering Environment Framework*. Tech Report, Carnegie Mellon Software Engineering Institute. (CMU/SEI) Technical Report.
38. Tilley, S., Huang, S. (2001). *Evaluating the Reverse Engineering Capabilities of Web Tools for Understanding Site Content and Structure: A Case Study*. Proc. 23rd International Conference on Software Engineering, IEEE, pp514-523.
39. Tonella, P., Ricca, F., Pianta, E. and Girardi, C. (2002). *Restructuring Multilingual Web Sites.* Proc. 18th IEEE Int. Conf. on Software Maintenance (ICSM'02), pp290-299.
40. Tonella, P., Torchiano, M., Bois, D. B., Systa, T. (2007). *Empirical Studies in Reverse Engineering: State of the Art and Future Trends.* Empirical Software Engineering, Springer Science & Business Media [2007].
41. Tramontana, P. (2005) *Reverse Engineering Web Applications*. Proceedings 21st International Conference on Software Maintenance (ICSM'05), IEEE, pp705-708.
42. Vanderdonckt, J., Bouillon, L. and Souchon, N. (2001). *Flexible reverse Engineering of Web Pages with VAQUISTA.* Proc 8th Working Conference on Reverse Engineering, WCRE'01, IEEE, pp241-248.