An Approach for Reverse Engineering of Web Applications

Sun Weijun[1,2] Li Shixian[1] Liu Xianming[3]
[1]Department of Computer Science, SUN YAT-SEN University
Guangzhou, 510275, China
[2]Faculty of Computer, Guangdong University of Technology
Guangzhou, 510006, China
[3]Department of Technology, JiangXi Electric Information and Communication Company
Nanchang, 330077, China
gdutswj@gdut.edu.cn, lnslsx@mail.sysu.edu.cn, ecopnlab@yahoo.com.cn

Abstract—**Web applications are the legacy software of the future. The Web application reverse-engineering process becomes necessary in order to facilitate the maintenance and evolution. This paper presents an approach to recover the architecture of web applications. The approach generates UML models from existing web applications through static and dynamic techniques. UML diagrams are extracted to depict the static, dynamic and behavioral aspect of Web applications. Finally, the architecture of the tool is described.**

Keywords-Web application; reverse engineering; UML; legacy software

I. INTRODUCTION

With the growth of the World Wide Web, more and more applications are developed using web technologies. Web applications have become the core business for many companies in several application areas such as on-line services (online retail, e-trading, e-banking, stock market, and so on). Such Web applications therefore act as valuable assets for companies and organizations.

Unfortunately, as a result of tight development schedules, a sound software development lifecycle and well-proven principles are not complied with in most cases of Web application development. As consequence, the documentation associated with many web applications is rarely complete or up-to-date, and the original developers of a maintained web application are often no longer part of the organization. This lack of documentation and the original developers increases the cost and time needed to understand and maintain large web applications.

Web Applications will be the next generation of legacy applications. These applications are required to undergo a reverse engineering process for the improvement of maintenance, evolution, migration and understanding.

Web applications are highly interactive and dynamic. A reverse engineering process should support the recovery of both the static and dynamic aspects of the applications, and suitable representation models should be used to render the recovered information.

This paper proposes and realizes an approach to assist developers in understanding existing web applications. A representation based on UML diagrams is extracted to depict the static, dynamic and behavioral aspects of Web applications.

The rest of this paper is organized as follows. Section 2 is the review of existing works in reverse engineering for web applications. Section 3 details web application modeling. Section 4 describes our reverse engineering approach for web applications. Section 5 describes the architecture of the reverse engineering tool. Finally, section 6 draws conclusions from this work.

II. RELATED WORK

Reverse engineering of Web Application is quite a recent field. Antoniol et al. in [1] proposed an approach, based on the Relational Management Methodology (RMM), to recover web site architectures. Antoniol et al. also proposed a tool, named WANDA,for Web application dynamic analysis[2]. WANDA aims to recover fine-grained details that allow a more accurate reconstruction of the Web application architecture and dynamic behavior. Ricca and Tonella have developed a semi-automatic tool named ReWeb[3,4,5], for reverse engineering Web applications into UML model, it performs several traditional source code analyses, and uses UML class diagrams to represent components and navigational features. Ricca and Tonella also proposed to enhance the analysis considering dynamic information[6]. Di Lucca et al. proposed an approach [7] and, then, a tool, named WARE[8,9], to recover Conallen's UML documentation from Web Application. Conallen's extensions presents the Web Application Extension (WAE) for UML[10].WARE performs static analyses on Web Application, stores the extracted information into a database and then uses such an information for the reverse engineering of UML diagrams. Dynamic analysis Web Application used in [11] to complement static information required to detect page clusters.

Other Web application reverse engineering approaches and tools are available in literature. Hassan and Holt[12], proposed a tool for Web application architecture recovery. Three layers in the Web applications (presentation, business and infrastructure) are identified through static analysis. WebUml[13] is a tool to reverse engineering Web application through dynamic analysis. Vanderdonckt et al. proposed a tool, named Vaquista[14], for reverse engineering of the presentation model of Web Application. Benslimane et al. [15,16]propose an approach known as 'OntoWeR' (Ontology based Web Reverse-engineering), The objective of OntoWeR is to enable conceptual schemas to be created.

III. WEB APPLICATION MODELING

Web applications are based on many components that are linked together to accomplish the functionality of the application. These components are often written in many different programming languages and potentially distributed over the Web.

Fig. 1 shows the metamodel of Web application structure (similar to [10], and like [13] model). The metamodel is used to describe the structure and components of a generic Web application. The core of a Web application is the *WebPage*, it can be static (*Client Page*) or dynamic (*Server Page*).A static page is a simple markup language (HTML) file on a Web server, its content is fixed and stored in a persistent way. A dynamic page contains a mixture of HTML tags and executable codes, and its content is generated by a Web server upon request of Web clients. Involved technologies are ASP, PHP, JSP, CGI and so on. When a dynamic page is requested, the application server preprocesses it and integrates data from various resources such as web objects or databases, to generate the final HTML web page sent to the browser.

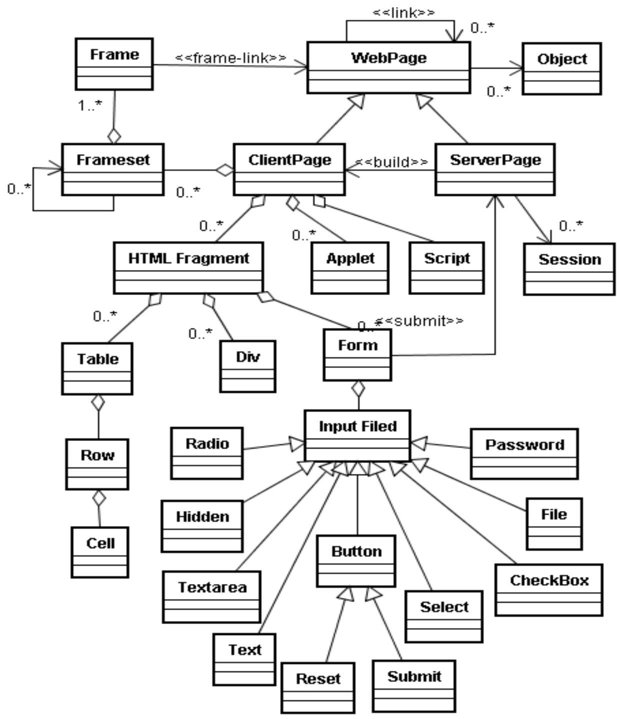

Figure 1. Web application structure metamodel

A Client page may be divided into frames using a particular page structure, the frameset. An HTML page may be inserted in a frame context, or can contain a set of frames grouped in a *Frameset* that can interact with each other. An HTML frame (Frame) is an area in the HTML client side page where the navigation can take place independently. In particular, a frame can be used to create menus defining navigation paths in other frames (framelink)[13].

The inner page components include *text, images, form, text box, multimedia objects* (sounds, movies)*, anchors* (implementing *hypertext links*)*, scripts*, and so on. Moreover, it can contain object types such as Java applets (*Applet*) or other embedded objects (*Object*) (e.g., ActiveX object, Microsoft COM object, and so on). Scripts (*VBScript and JavaScript*) and Java applets represent page active components, since they perform some processing action that contributes to the Web application behavior. Client side scripting code fragments (*Script*) supports dialogue with the user (such as alerts, inputs boxes, and so on) and functions declarations. The scripting code may use a Cookie [13,18].

A *ServerPage* defines a server side page composed by variables, prompts, alerts, an so on (class attributes), and function declarations (class methods). A server page can use embedded objects (*Object*) and can define sessions (*Session*). A server page may receive data from (*Form*) elements, it may contain redirection links or may build a client side page based on (*Form*) data. *HTMLFragment* defines HTML code fragments dynamically built by server pages [13,18].

The behavioral and the navigational structures of a Web application are generally achieved through collaborations and interactions between its structural components. During the execution of an instrumented Web application, the sequences of interactions implementing each behavior will have to be identified in the code and represented with suitable models too.

To this aim, a Web application is modeled as a set of web pages that a user can access sequentially along a working session [17]. Fig. 2 shows the view of a Web application (similar to [2] and [18]).

An accessed page may be a *Server Page* or a *Client Page*, by which the user interacts with the Web application. A *Transition* consists of sequentially visited pages by navigating a link from a Source Page to a Target Page. A transition is categorize into different types of relationships between pages *(Hyperlinks, form Submission, Build, Redirection, Inclusion*).Thus, all the executions of the Web application is represented by the *Execution Trace*, which involves many single *User Session Trace* of navigated pages[17].

Web applications can be described in UML with various diagrams: class diagrams for the structure and components of a Web application, state diagrams for component states and state changes, collaboration diagrams for scenarios and interactions among objects, use-case diagrams for application functionalities and interactions with external systems.

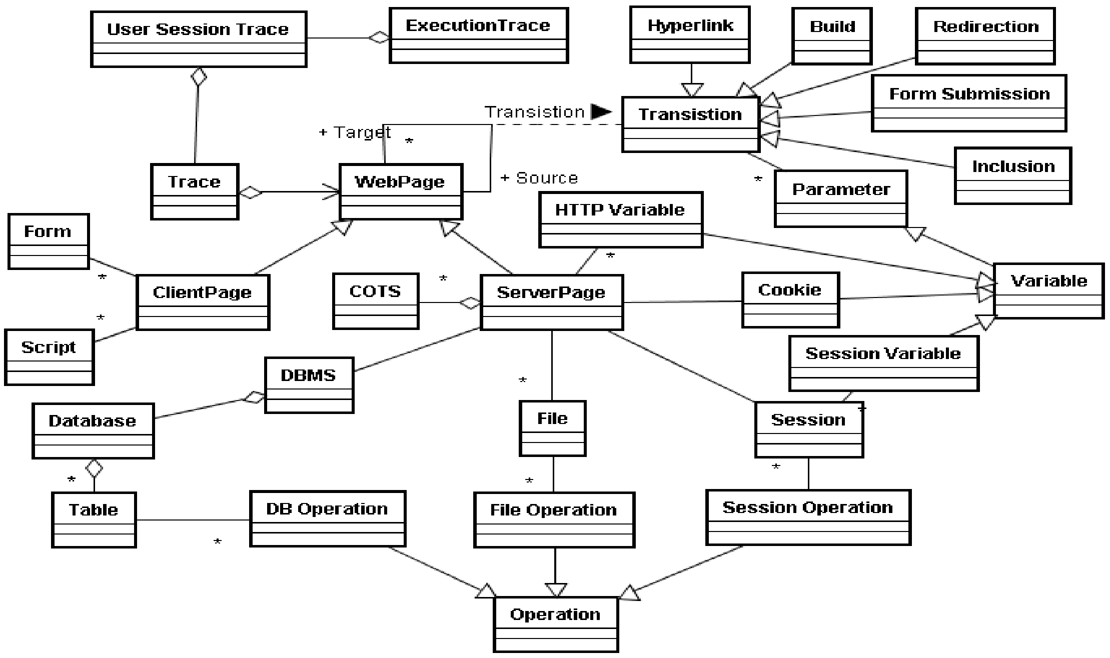

Figure 2. Web application dynamic model

IV. Reverse Engineering Approach

As Fig. 3 shows, the approach consists in three successive phases: Analysis, Abstraction, Formatting and Visualization.

The analysis phase is responsible for parsing and instrumentation of web applications. The abstraction phase focuses on the Model abstraction. The formatting and visualization phase is responsible for the Result presentation.

Next, the phases of the Web application reverse engineering approach are presented in detail.

A. Analysis

The analysis phase is performed into three steps: preprocessing, static analysis, and dynamic analysis. This resulting information is stored in a repository.

1) Preprocessing

As HTML is a short-constrained language, the validity of web pages should be check before working on it. The preprocessing phase takes web pages as input, and corrects them, proceeds to some filtering and cleaning, executes some transformations and then returns a DOM describing the page structure for each page. Filtering and cleaning consists in checking the source code of HTML pages, eliminate useless tags such as those of layout (e.g. <b>, <i>), and preserve useful tags, which carry information to be treated in the following stages (e.g. <form>, <table>, <td>, <tr>, <ul>, <li>)[15]. The result of this step is a set of cleaned HTML pages. Some Tools allow to transform any HTML page into a well-formed XHTML-valid document[19]. The result is an XML document that can be parsed and transformed by specific processors such as DOM[20] or XSLT[21].

It is useful to gather Web pages within a site into semantic groups according to their informational content. A page type is a set of pages related to a same concept. All the pages of a page type defined as a set of pages related to a same concept have some similarities: they display the same pieces of information and have a very similar, while possibly different, layout. The file path is a good clue to detect the page types through a web site [22].

2) Static Analysis

Static analysis is based on a parser that analyzes source code of web applications. The parser generates facts about the components, relations and attributes of the Web applications. Because of the mix of languages (HTML, JavaScript, VBScript, Java, embedded objects and so on),the parser should not depend on a single extractor. To deal with the web application, five types of extractors are used in this stage.

The five types of extractors consist of HTML extractor, Server Script extractor, DB Access extractor, Source Code extractor and Binary Code extractor. Each extractor parses a component or a section within a web page and generates the appropriate facts. Together these extractors generate facts from the entire web application. The parser crawls the directory tree of the source code and invokes the corresponding extractor according to the type of the component.HTML extractor performs different types of analysis including the text analysis, structure analysis, style analysis, augmentation analysis, and media analysis. Structure analysis includes table hierarchy analysis, frame analysis, link Analysis and form analysis.

3) Dynamic Analysis

Dynamic analysis is based on the information recorded during the executions of an instrumented Web Application.

The instrumentation of the Web Application is obtained by using the tool WANDA [2] that automatically instruments the source code of a Web Application by inserting probes. The probes are able to identify relevant dynamic information. Probes are developed to collect dynamic information such as page access, HTTP environment variables, cookie and session management functions, databases and file I/O, access to libraries and COTS and so on. The dynamic model of Web Application is shown in Fig. 2.

A Web Application execution trace is composed of a sequence of navigated pages. To collect execution traces useful for an effective extraction of UML models, the instrumented Web Application needs to be executed in a real usage environment for the collection of information about the interaction of users with the Web Application.

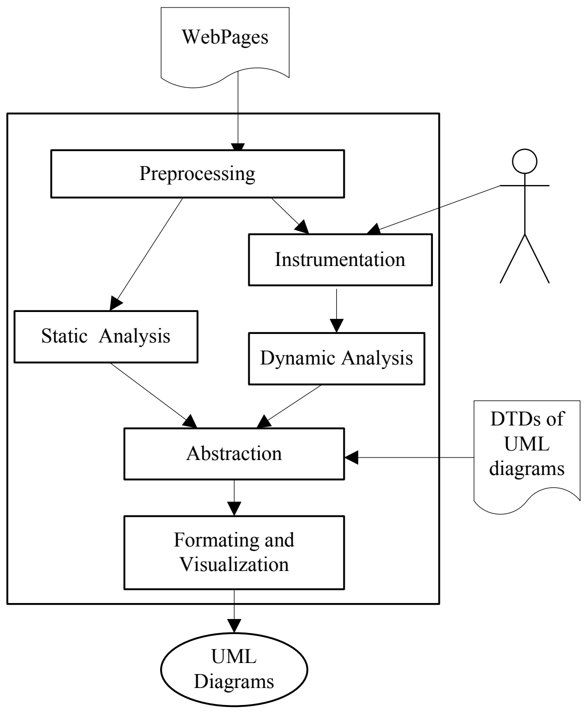

Figure 3. Reverse engineering process

B. *Abstraction*

The phase implements the abstraction operations that are necessary for producing more abstract views from the Web Applications. The abstraction works by retrieving static and dynamic information about the web application, such as the list of the page hyperlinks, page components, page or form parameters, functions activated in a page, and sequences of function calls or link activations. The results of the queries are summarized to create diagrammatic representations. Queries are mostly conceived to detect the building blocks of the Web application (to extract its architecture) and their temporal sequence of interaction (to extract sequence diagrams). Moreover, the interaction frequency (e.g., the frequency of exercising an association), the information exchanged between the entities (e.g., passed parameters, state values stored in session data, read and writing from/to files or database) are exploited to significantly enhance the UML representation.

C. *Formatting and visualization*

With the information abstracted by the previous phase, the model describing the structure of a Web application at different degrees of detail can be built. The proposed model extends the one of Conallen's extentions. Class diagrams are used to describe frames, Java applets, input fields, cookies, scripts, and so on),while state diagrams are used to represent behavior and navigational structures (client-server pages, navigation links, frames sets, inputs, scripting code flow control, and so on).

To facilitate the transformation in any possible visualization format, the abstracted diagrams are stored in XML consistent with the XML Metadata Interchange (XMI).

V. Architecture of the Tool to be implemented

The tool to be implemented is composed of three subsystems to allow assuming the stages of Web application reverse-engineering approach, as well as supporting interaction with the user (Fig. 4).

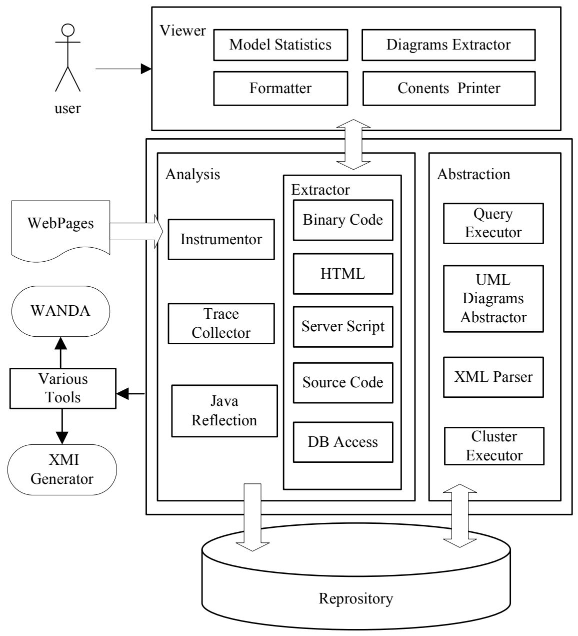

Figure 4. Architecture of the tool to be implemented

- Analysis

Analysis module allows the preprocessing, static analysis and dynamic analysis of pages of Web application. Source codes of Web pages are checked and the components of the software system are processed using specialized extractors. The instrumented Web application is executed for collecting dynamic information.

- Abstraction

Abstraction module represents an implementation of the abstraction phase in the reverse-engineering process. It allows producing more abstract views from the Web Applications by retrieving static and dynamic information.

- Viewer

Viewer allows viewing the resulting UML diagrams.

VI. Conclusions

In this paper, a reverse engineering approach has been proposed for abstracting UML diagrams from Web Applications. These UML diagrams deal with not only static content, but also with the more challenging dynamic content of Web Applications. These diagrams, together with other extracted documentation, constitute an important support for the subsequent maintenance and evolution of Web Applications.

The proposed reverse-engineering approach consists in three phases: analyses, abstraction, formatting and visualization. The approach allows users to recover static and dynamic Web sites. Some of the required reverse engineering activities are automatically performed by tools, while other activities are carried out semi-automatically, with the assistance provided by tools. Our further work is devoted to improving the reverse engineering approach and tools supporting.

Acknowledgment

Supported by the National Natural Science Foundation of China under Grant No. 60673122, and by the Natural Science Foundation of Guangdong under Grant No. 8151030007000002, and by the Technology Foundation of Jiangxi Electric Power Corporation under Grant No.200850802.

References

[1] G. Antoniol, G. Canfora, G. Casazza, "Web site reengineering using RMM," Proc. International Workshop on Web Site Evolution, March 2000, pp. 9-16.

[2] G. Antoniol, P. M. Di and M. Zazzara, "Understanding Web applications through dynamic analysis," Proc. 12th IEEE International Workshop on Program Comprehension, 2004., pp. 120-129

[3] F. Ricca. P. Tonella, "Web site analysis: Structure and evolution," in Proceedings of IEEE International Conference on Software Maintenance,ICSM 2000, pp. 76-85.

[4] F. Ricca. P. Tonella, "Analysis and testing of web applications," in Proceedings of the International Conference on Software Engineering, ICSE 2001, pp. 25-34.

[5] F. Ricca and P. Tonella, "Understanding and restructuring Web sites with ReWeb," Multimedia, IEEE, vol.8, pp. 40-51, 2001.

[6] F. Tonella, P. Ricca, "Dynamic model extraction and statistical analysis of Web applications," Proc. Fourth International Workshop on Web Site Evolution, 2002, pp. 43-52.

[7] G.A Di Lucca, P. M. Di, G. Antoniol and G. Casazza, "An approach for reverse engineering of web-based applications," Proc. Eighth Working Conference on Reverse Engineering, 2001,pp. 231-240.

[8] G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana and C. U. De, "WARE: a tool for the reverse engineering of Web applications," Proc. Sixth European Conference on Software Maintenance and Reengineering, pp. 241-250.

[9] G. A. Di Lucca, A. R. Fasolino and P. Tramontana, "Reverse engineering Web applications: the WARE approach," Journal Of Software Maintenance And Evolution-Research And Practice, vol.16, pp. 71-101, 2004.

[10] J. Conallen, Building Web Applications with UML(2nd Edition), Addison-Wesley Publishing Compay, 2002.

[11] G. A. Di Lucca , A. R. Fasolino and P. Tramontana, "Towards a better comprehensibility of web applications: lessons learned from reverse engineering experiments," Proc. Fourth International Workshop on Web Site Evolution, 2002,pp. 33-42.

[12] A. E. Hassan and R. C. Holt, "Architecture recovery of Web applications," Proc. the 24rd International Conference on Software Engineering, ICSE 2002,pp. 349-359.

[13] A. T. Carlo Bellettini,Alessandro Marchetto, "WebUml : reverse engineering of web applications," Proc. the ACM symposium on Applied computing, 2004,pp. 1662-1669.

[14] J. Vanderdonckt, L. Bouillon and N. Souchon, "Flexible reverse engineering of web pages with VAQUISTA," Proc. Eighth Working Conference on Reverse Engineering, 2001,pp. 241-248.

[15] M. A. B. Bouchiha, D., Malki, M., "Ontology based Web Application Reverse-Engineering Approach," INFOCOMP journal of Computer Science, vol.6(1), pp. 37-46, 2007.

[16] B. Benslimane, M. S., Malki, M., Bouchiha, D., "An Ontology Based Web Application Reverse Engineering Approach," International Review on Computers and Software, 1(1), pp. 52-58, 2006.

[17] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, "Recovering Interaction Design Patterns in Web Applications," Proc. Ninth European Conference on Software Maintenance and Reengineering, CSMR 2005, pp. 366-374.

[18] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, "Supporting Web application evolution by dynamic analysis," Proc. Eighth International Workshop on Principles of Software Evolution,2005,pp. 175-184.

[19] World Wide Web Consortium: "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) ",http://www.w3.org/TR/xhtml1/

[20] World Wide Web Consortium: "Document Object Model(DOM) ", http://www.w3.org/DOM/

[21] World Wide Web Consortium: "XSL Transformations(XSLT) ", http://www.w3.org/TR/xslt

[22] E. Fabrice, F. Aurore, H. Jean, H. Jean-Luc, "A tool-supported method to extract data and schema from web sites". Proc. the fifth international workshop on Web site evolution, 2003, Amsterdam, pp. 3-11.