Vision-based Presentation Modeling of Web Applications: A Reverse Engineering Approach

Natheer Khasawneh
Department of Software Engineering, Jordan University of Science and Technology, Irbid, Jordan
Email: natheer@just.edu.jo

Oduy Samarah
Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan
Email: oasamarah09@cit.just.edu.jo

Safwan Al-Omari
Department of Software Engineering, Jordan University of Science and Technology, Irbid, Jordan
Email: ssomari@just.edu.jo

Stefan Conrad
Institute of Computer Science, Heinrich Heine University, Dusseldorf, Germany
Email: conrad@cs.uni-duesseldorf.de

***Abstract*—Presentation modeling, which captures the layout of an HTML page, is a very important aspect of modeling Web Applications (WAs). However, presentation modeling is often neglected during forward engineering of Web Applications; therefore, most of these applications are poorly modeled or not modeled at all. This paper discusses the design, implementation, and evaluation of a reverse engineering tool that extracts and builds appropriate UML presentation model of existing Web Applications. The tool consists of three steps. First, we identify and extract visual blocks and presentation elements of an HTML page such as navigation bars, header sections, text input, etc. In this step, we adopt the VIPS algorithm, which divides an HTML into semantically coherent blocks. Second, the identified presentation elements in step one are mapped to the most appropriate UML presentation model elements. Third, the resulting presentation model is made available in Magicdraw for manipulation. Our approach is applied and evaluated in the Goalzz home page.**

***Index Terms*—Reverse Engineering, Web Application, Web UML, Vision-based Page Segmentation**

I. Introduction

Recently, many applications and services have evolved from being stand-alone and monolithic applications into web applications. According to a recent study [1], most of the existing web applications lack proper modeling, which is necessary for maintenance, reengineering, and proper evolution to emerging web technologies. For the purpose of modeling legacy web applications, there is an urgent need to have a reverse engineering method to extract models of existing web applications. Chikofsky describes 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" [2].

Current modeling languages and methodologies are not sufficient for capturing all aspects of web applications. UML for example is not sufficient to express the hyperlinks between different HTML pages. Modeling of web applications can be performed at three levels:

1. Content modeling: focuses on modeling data in an HTML page.
2. Hyper text modeling: focuses on modeling links between HTML pages in a web application.
3. Presentation modeling: focuses on the layout of the items inside a particular HTML page.

In this paper we focus on the presentation model, which is used to model the page layout in a UML presentation model. The approach we follow in generating a presentation model is based on page segmentation method discussed in [3]. Page segmentation divides the page into different blocks according to its visual appearance when rendered in a web browser.

The reset of the paper is organized as follows: in Section 2, we provide an overview of the related work; Section 3 introduces the proposed approach in details; Implementation details are discussed in Section 4; Section 5 discusses case studies to illustrate and demonstrate our approach; finally in Section 6, we sketch concluding remarks and future work.

II. Related Work

In the literature there are many methods and tools of web reverse engineering which are built on the standard of reverse engineering techniques. These methods and tools can be used to describe and model the web

© 2012 ACADEMY PUBLISHER
doi:10.4304/jetwi.4.2.134-141

applications with respect to different levels: content, hypertext, and presentation [4].

The UML modeling language is the most widely used during the forward engineering design process and in many web application reverse engineering techniques. For example, in [1] authors defined a process for reverse engineering by describing a method for understanding web applications to be easily maintained and evolved. In [5], authors showed how web applications with UML presentation can be easily maintained. The approach in [6] is based on structured and model based techniques. In this approach the HTML page is divided into several blocks according to a cognitive visual analysis. After that the specific patterns with these blocks are extracted to produce structural blocks and through these structural blocks a conceptual model is represented. The approach in [7] relies on HTML pages analysis by extracting the useful information from the web page and analyze the extracted information using the domain ontology and form the analysis results the UML conceptual schema is generated.

III. Approach and Methodology

In this paper, we present a new approach to reverse engineer existing web applications into UML presentation model. The proposed approach focuses on discovering the structure of the web page and presenting the structure in UML presentation model, as shown in Fig. 1.

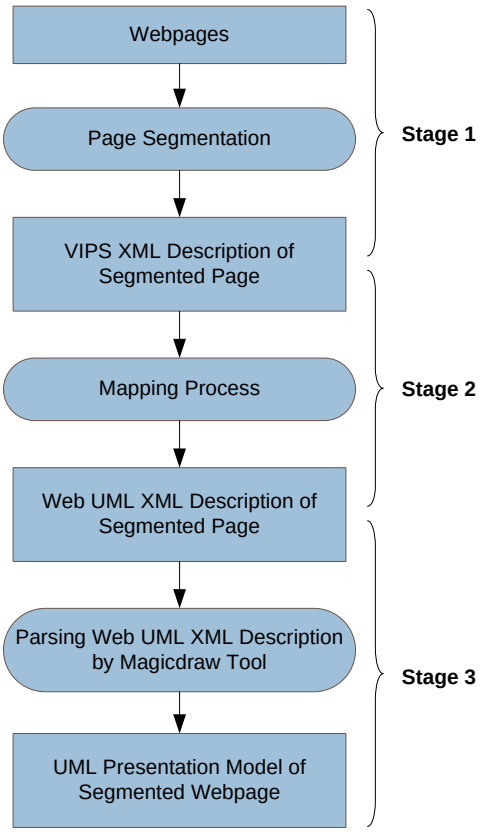

Figure 1. Approach Architecture.

The presented method consists of three stages: page segmentation stage, mapping process stage, and UML model generation stage. Page segmentation stage accepts an HTML page as input and produces an XML description of segmented page. Mapping process stage transforms the XML description to Web UML XML description. Finally, UML generation Stage accepts a Web UML XML description of the segmented page and outputs the UML presentation model.

In the rest of this section, we describe, in details, page segmentation stage, web UML XML, mapping process stage, and UML model generation stage, respectively.

A. Page Segmentation Stage

In this stage we use Vision-based Page Segmentation Algorithm (VIPS) [3] to segment the web page into different blocks. VIPS incorporates both the page appearance (visual cues) and Document Object Model (DOM) to perform the segmentation.

VIPS is done in three steps (Fig. 2): block extraction, separator detection, and content structure construction. These steps are repeated recursively several times until a user-defined threshold is reached. The threshold is called Permitted Degree of Coherence (PDoC) and it is based on Degree of Coherence (DoC). DoC is a numeric value between 1 and 10 that increases as the consistency between blocks increases. Following is a description of each step in VIPS:

Visual Block Extraction

The input to the visual block extraction is the visual

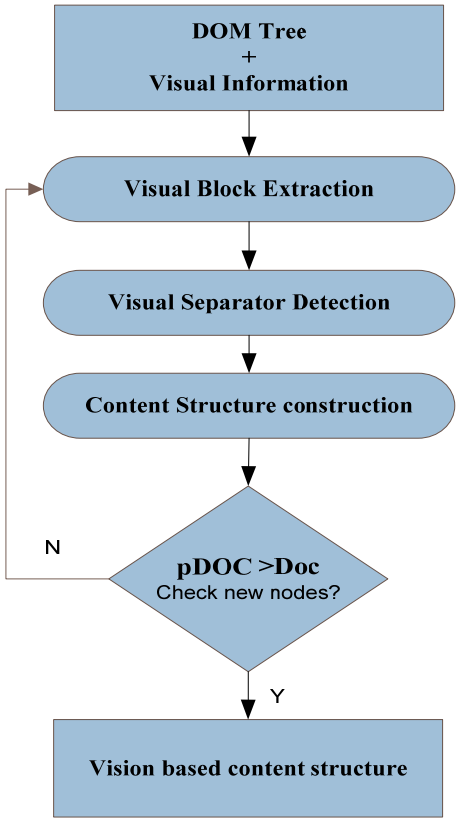

Figure 2. Flowchart of the Segmentation Process.

cues and the DOM tree of the web page. During this step each node in the DOM tree is matched with the block where it belongs to in the visual cues. From the tree node,

© 2012 ACADEMY PUBLISHER

VIPS starts looking if the sub nodes belong to the same block. Nodes with a coherence value less than PDoC are matched together to belong to the same group. This process is repeated recursively for the sub roots of the unmatched nodes until all nodes are matched to a block in the visual cues.

Visual Separator Detection

The input to this step is the collection of the extracted blocks from the previous step, whereas, the output is a set of separators which separate different blocks. There are two types of separators, horizontal separator and vertical separator. The process starts by one separator which spans the whole page. Then blocks are added one by one and the separator gets updated according to the following rules:

1. If the added block falls inside the separator, the separator will be splitted into two. The splitting will be done either vertically or horizontally.
2. If the added block covers part of the separator the separator will be resized.
3. If the added block covers the whole separator, the separator will be removed.

Weight separator is assigned to each separator by considering factors that show how similar the neighboring blocks are. For a separator which separates two blocks, the more the two blocks differ the higher the weight that will be assigned for the separator and vice versa. The used factors are: distance between blocks, overlapping with HTML tags, background differences, font differences, and structure similarity.

Content Structure Construction

In this step content structure is constructed by merging lowest weight separators with the neighboring separators. The newly merged separator will be given a DoC value equals to the maximum DoC of the merged separators. This step is iterative until a separator with maximum weight is reached. Finally, each node in the newly generated block is checked whether its DoC meets the condition given by PDoC or not. If not, the Visual Extraction Process starts over. Otherwise, the process terminates and the page structure is generated in VIPS XML description format.

VIPS XML description format is the immediate output of the VIPS algorithm. It is very simple and understandable description, which is defined for blocks and their content in the segmented page. The VIPS XML description captures and describes some attributes for each block and HTML elements in a web page, as shown in Table I. This table contains the main attributes which are used during the mapping process stage.

Fig. 3 represents the VIPS XML description for blocks with ID (1-2) which contains three elements irrespective of the type of these elements (block or primitive HTML Element). Fig. 3 also shows more information for block such as coordinates and not containing any tables or images. In contrast, Fig. 4 represents VIPS XML description for primitive HTML Element (anchor) with ID (1-2-1), and also Fig. 4 shows more information about the element such as coordinates and Uniform Resource Locator (URL) for anchor in SRC attribute. Through ID attribute, we can know that the block with ID (1-2) contains the HTML element with ID (1-2-1).

TABLE I.
VIPS XML ATTRIBUTES AND THEIR DESCRIPTION

VIPS XML Attributes	Descriptions
ContainImg	Determine whether the block contains image
IsImg	Determine whether the HTML Element is image
ContainTable	Determine whether the block contains table
ContainP	Determine whether the HTML Element contains text
TextLen	Determine the length of text in blokes or HTML Elements
DOMCldNum	Determine whether the element is block or primitive HTML tag (it is block if DOMCldNum >1)
ObjectRectLeft ObjectRectTop ObjectRectWidth ObjectRectHeight	Determine the coordinates for blocks or HTML Elements
Content	Determine the content of blocks and anchor which appears for user .(used for data mining techniques)
SRC	Determine the source of image, anchor or any HTML Element
ID	Unique ID for block or HTML Element, which is used for specific purpose during the implementation.
order	Unique ID (integer number)

<LayoutNode FrameSourceIndex="0" SourceIndex="16" DoC="10" ContainImg="0" IsImg="false" ContainTable="false" ContainP="0" TextLen="15" LinkTextLen="15" DOMCldNum="3" FontSize="7.5" FontWeight="700" BgColor="#00ffff" ObjectRectLeft="10" ObjectRectTop="289" ObjectRectWidth="202" ObjectRectHeight="236" ID="1-2" order="10">

Figure 3. VIPS XML Description for Block.

<LayoutNode FrameSourceIndex="0" SourceIndex="18" DoC="11" ContainImg="0" ContainTable="false" ContainP="0" TextLen="5" LinkTextLen="5" DOMCldNum="1" FontSize="7.5" FontWeight="700" BgColor="#00ffff" IsImg="false" ObjectRectLeft="60" ObjectRectTop="308" ObjectRectWidth="95" ObjectRectHeight="36" Content="Page1 " SRC="<A href= " FirstPage. html" >Page1</A> " ID="1-2-1" order="11"/>

Figure 4. VIPS XML Description for Anchor HTML Elements.

© 2012 ACADEMY PUBLISHER

B. Web UML XML

Web UML XML description is based on UML-Based Web Engineering (UWE) Metamodel [8]. UWE is a methodology used for Web application modeling purpose, especially, structure modeling and behavior modeling. Furthermore, this methodology provides guidelines for systematic modeling of Web applications. UWE comprises four main parts to model web application: Notations, Methods, Metamodel, and Process.

Next we present a brief description about the UWE notation before discussing the process of transforming VIPS XML description format to Web UML XML format.

UWE Notation

For UML presentational model, UWE defines stereotypes for user interface elements, these elements are divided into two parts: primitive user interface (UIElement) elements and user interface container (UIContainer) elements which contain a collection of primitive user interface elements.

A case-tool Magicdraw supports the design and model of web applications with different aspects by using UWE notations and Metamodels as plug-in. This case-tool is used to parse the Web UML XML description to generate UML presentation model in generation model stage.

Table II summarizes the UWE Stereotypes, which are used to model web application into UML presentation model. These Stereotypes are:

1. Page: A Page element is the area of the user interface which contains all of UIElement elements and UIContainer elements. The page constructs the root of presentation model.
2. Anchor: An anchor element permits the user to move from page to another page, or from location to another location on the same page.
3. Group Presentation: A group presentation element is used to define a set of UIElement elements, such as a collection of anchors.
4. Text: A text element is used to display a sequence of characters.
5. Textinput: A text input element permits the user to enter text.
6. Form: A form element contains a collection of UIElement elements that are used to provide data for a submitted process.
7. Button: A button element allows the user to initiate some actions on the web page. Actions include submitting the content for Textinput element, playing video, displaying image, triggering anchor and so on.
8. Selection: A selection element displays a list of items for the user to select one or more items.
9. Image: An image element is used to display the image.
10. Media Object: MediaObject elements are used to play multimedia objects such audio and video.

TABLE II.
UWE SEROTYPES SYMBOL.

Stereotypes	Symobl
Page	
Anchor	
Text	
Text input	
Selection	
File upload	
Image	
mediaObject	
Form	
Button	
Collection	
Custom component	

11. File upload: A File upload element allows user to upload files.
12. Custom component: UWE also defines custom component stereotypes for custom HTML elements which are not defined by UWE.

The Web UML XML description is a complicated description which is used to describe the UWE stereotypes in XML format. For that, this type of XML description can be interpreted and rendered to UWE stereotypes by any case-tool that is capable of parsing this XML description to build a UML presentation model.

The Web UML XML description is divided into three main parts, each part specifies some properties for elements. Fig. 5, Fig. 6, and Fig. 7 contain the UML XML description for a simple block of web page, this block is represented by the DIV HTML tag which consists of two Textinput elements and one submit button as shown in Fig. 8.

The Web UML XML description in Fig. 5 specifies the ID number and name for each element, and also determines whether the element is visible or not in UML presentation model. The benefit of visibility property appears through modeling the hidden HTML elements that are not displayed by the browser.

```
<packagedElement xmi:type='uml:Class' xmi:id='1_1_2'
name='Administrator Login' visibility='public'>
<ownedAttribute xmi:type='uml:Property' xmi:id='5'
name='textinput' visibility='private' aggregation='composite'
type='1_1_2_1'/>
<ownedAttribute xmi:type='uml:Property' xmi:id='6'
name='textinput' visibility='private' aggregation='composite'
type='1_1_2_2'/>
<ownedAttribute xmi:type='uml:Property' xmi:id='7'
name='button' visibility='private' aggregation='composite'
type='1_1_2_3'/>
</packagedElement>
--------------------------------------------------------------------------------
<packagedElement xmi:type='uml:Class' xmi:id='1_1_2_1'
name="Username' ' visibility='public'/>
<packagedElement xmi:type='uml:Class' xmi:id='1_1_2_2'
name="Password' visibility='public'/>
<packagedElement xmi:type='uml:Class' xmi:id='1_1_2_3'
name='Submit' visibility='public'/>
```

Figure 5. UML XML Description (Part1).

© 2012 ACADEMY PUBLISHER

Fig. 5 is also divided into two parts as shown by the dashed lines. In the first part, the block is defined as a class type and the elements in this block are defined as a property type. In the second part, the elements in block are only defined as a class type.

This means that the group element, or in other words UIContainer elements, is defined as a class type only, and the primitive user interface (UIElement) element is defined as a class and property type. Fig. 6 represents the second part of Web UML XML description. In this part, the coordinates for each element are specified by four points as shown in bold text in Fig.7. These points are:

1. Padding-left: sets the left padding (space) of an element.
2. Padding-Top: sets the top padding (space) of an element.
3. Width: sets the width for element.
4. Height: sets the height for element.

These points are structured as: Padding-left, Padding-Top, Width, and Height.

```
<mdElement elementClass='Class' xmi:id='1004'>
 <elementID xmi:idref='1_1_2'/>
 <properties>
  <mdElement elementClass='BooleanProperty'>
<propertyID>SUPPRESS_STRUCTURE</propertyID>
<propertyDescriptionID>SUPPRESS_STRUCTURE_DESCRIP
TION</propertyDescriptionID>
</mdElement>
</properties>
<geometry>90,225, 20, 130</geometry>
<propertyID>SUPPRESS_STRUCTURE</propertyID>
<compartment xmi:value='5^6^7'
compartmentID='ATTRIBUTES'/>
<parts>
<mdElement elementClass='Part' xmi:id='1005'>
<elementID xmi:idref='5'/>
<geometry>100,255, 20, 20</geometry>
<propertyID>SUPPRESS_STRUCTURE</propertyID>
</mdElement>
<mdElement elementClass='Part' xmi:id='1006'>
<elementID xmi:idref='6'/>
<geometry>100,280, 20, 20</geometry>
<propertyID>SUPPRESS_STRUCTURE</propertyID>
</mdElement>
<mdElement elementClass='Part' xmi:id='1007'>
<elementID xmi:idref='7'/>
<geometry>100,305, 20, 20</geometry>
<propertyID>SUPPRESS_STRUCTURE</propertyID>
</mdElement>
</parts>
</mdElement>
```

Figure 6. UML XML Description (Part2).

Fig. 7 represents the third part of Web UML XML description. In this part, the stereotype of element is specified by UWE profile. Also, Fig. 7 is divided into two parts as shown by the dashed lines, First part specifies the stereotype of element from class type, and the second part also specifies the stereotype of element from property type, as is shown by bold text in Fig. 7.

C. Mapping Process Stage

The mapping process is considered the main part of our approach. This process accepts the VIPS XML description as input file and produces Web UML XML

```
<UWE_PROFILE:PRESENTATIONGROUP XMI:ID='1018'
BASE_CLASS='1_1_2'/>
<UWE_PROFILE: TEXTINPUT XMI:ID='1019'
BASE_CLASS='1_1_2_1'/>
<UWE_PROFILE: TEXTINPUT XMI:ID='1020'
BASE_CLASS='1_1_2_2'/>
<UWE_PROFILE:BUTTON XMI:ID='1021'
BASE_CLASS='1_1_2_3'/>
-------------------------------------------------------------------------------
<UWE_Profile: textInput xmi:id='1030' base_Property='5'/>
<UWE_Profile: textInput xmi:id='1031' base_Property='6'/>
<UWE_Profile:button xmi:id='1032' base_Property='7'/>
```

Figure 7. UML XML Description (Part3).

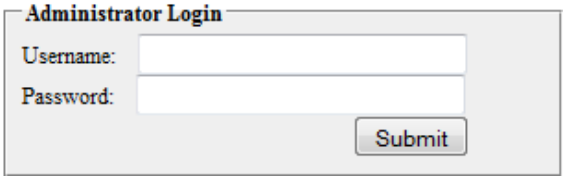

Figure 8. Group of HTML Elements.

description as output file. Substantially, the mapping process consists of the following steps:

1. Firstly, Contrast Map table between HTML elements and UWE stereotypes, as shown in Table III.
2. Read the VIPS XML Description for each block, this step continues until the end of VIPS XML Description file is reached.
3. The VIPS XML attributes for each element are examined, and then useful information from these attributes such as the type of element and coordinates are extracted.

TABLE III.
VIPS XML ATTRIBUTES AND THEIR DESCRIPTION.

HTML Element	UEW stereotype
  <fieldset></fieldset>	Group Presentation
<select> <input type=" radio"/> ☐	Selection
<button></button> <input type=" reset " /> <input type=" button " /> <input type=" submit " />	Button
<input type=" file " />	File upload
<input type=" text " /> <input type=" password " /> < textarea></ textarea>	Text input
	Anchor
	Text
<form > </form>	Form
<img src=""></img>	Image

4. The extracted information for all elements in block is stored together in the data structure list for block.
5. Examine the data structure list for each block to check whether the block has a collection of elements

© 2012 ACADEMY PUBLISHER

that have the same type or not. For example, check whether the blocks contain a collection of anchors or not.

6. Mapping location step is considered the most important step; this step needs to change the coordinates for elements to become consistent within UML presentation model.
7. Mapping from HTML element to UWE Stereotypes.
8. Now, the content of data structure list for each block is Web UML XML Description. This content is stored into Web UML XML file.

Fig. 9 shows the pseudo code for mapping process.

```
1: Input: VIPS XML description for blocks in the segmented Web Page.
2: Output: UML XML description for blocks in the segmented page.
3: Begin
4: Counter ← 1          // Counter for blocks.
5: total_blocks ← N // N is the no. of blocks in segmented page.
6: For counter to total_blocks step by 1
7: Block← list_of_blocks [counter]
8: Counter_elemnent ← 1          // Counter for elements in block
9: While Block contains elements with VIPS XML description
10: Element← Block [Counter_elemnent]
11:VIPS_ Info ← Extract useful information from VIPS XML att. of Element.
12: WebUMLXML_info ← Mapping _to_WebUMLXML (VIPS_ Info).
13: Block [Counter_elemnent] ← WebUMLXML_info
14: Counter_elemnent INCREMNET BY 1
15: End While
16: Store the information of block and elements into Web UML XML file.
17: End for
```

Figure 9. Mapping Process Pseudo Code.

D. UML Presentation Model Generation Stage

After mapping process stage, the Web UML XML description is used as input file for Magicdraw tool to generate UML presentation model. The UML presentation model in Fig. 10 is generated by parsing the WebUML XML Description in Fig. 5, Fig.7 and Fig. 7 which form the simple HTML block as shown in Fig. 8.

IV. Implementation Details

The process described in section 3 is implemented using C#.Net Programming language (.NET Framework 3.5). This language provides the object oriented programming which assists in programming process through providing the libraries which provides many features. In addition, we use the following APIs:

1. microsoft.mshtml.dll
2. MSXML3.dll
3. PageAnalyzer.dll

The PageAnalyzer.dll is considered the main API used in our implementation; this dll file generates the VIPS XML Description for any Web Page.

The main challenges which we faced during the implementation process are:

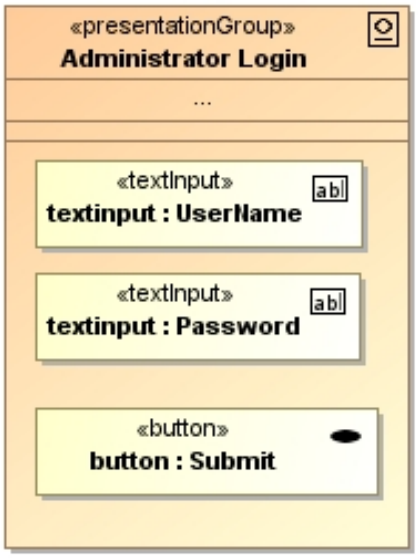

Figure 10. Simple UML Presentation Model.

1. Mapping the coordinates of element and block in VIPS XML Description to coordinates in Web UML XML Description.
2. Take into account all the HTML elements.

V. Case Study of the Proposed Approach

In this section, we will illustrate the reverse engineering process described earlier by applying it to a specific website.

A. Page Segmentation

Fig. 11 shows the home page for goalzz web site , this page is segmented using the VIPS algorithm into blocks as shown in Fig. 12, some of these blocks are small and others are large in size depending on the structure of the page. The page layout of the segmented page is shown in Fig. 13. In addition, Fig. 14 shows the DOM tree for the segmented page.

B. Mapping Process

After the page segmentation process, the VIPS algorithm assigns each block, sub-block, and element with XML descriptions; these descriptions capture some properties of each block and sub-block in the web page, then the VIPS XML format is mapped and transformed into the Web UML XML format as illustrated in Fig. 2.

Figure 11.Web Page before Segmentation Stage.

© 2012 ACADEMY PUBLISHER

C. UML Presentation Model Generation Stage

After the mapping and transformation process is achieved, the presentation model can be imported and manipulated in the Magicdraw modeling tool. Fig.15 shows the presentation model for goalzz home page.

Figure 12.Web Page after Segmentation Stage.

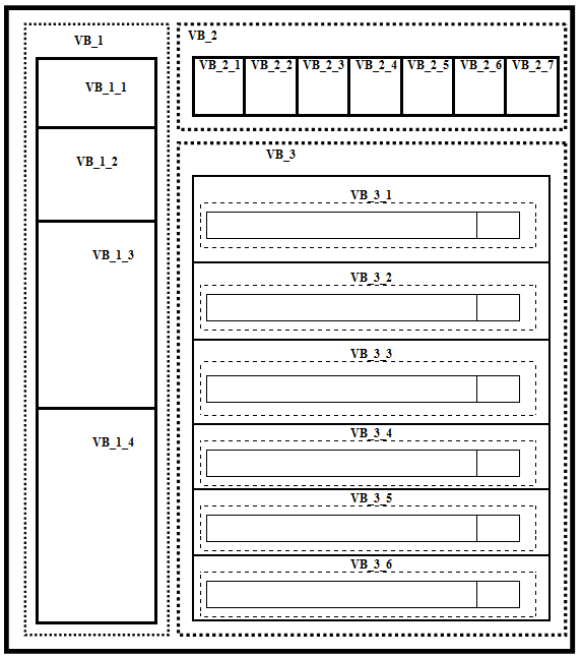

Figure 13. Page Layout for Goalzz Homepage.

VI. Summary and Future Work

In this paper we present an approach of reverse engineering web applications into UML web presentation model. This issue is evolved from the more generic reverse engineering process, concentrating on the structure of the web page. We have presented an approach to the identification of structure in a web page and model this structure in UML presentation model. The approach relies on a number of structured techniques such as page segmentation.

Future work will concentrate on building a complete framework which automatically builds the UML presentation model for any given application. The process of presenting the UML presentation model will be automated and will apply content mining along with the segmentation technique to accurately identify different blocks of the web page.

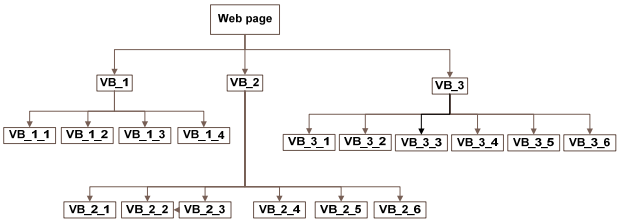

Figure 14. DOM Tree for Segmented Page.

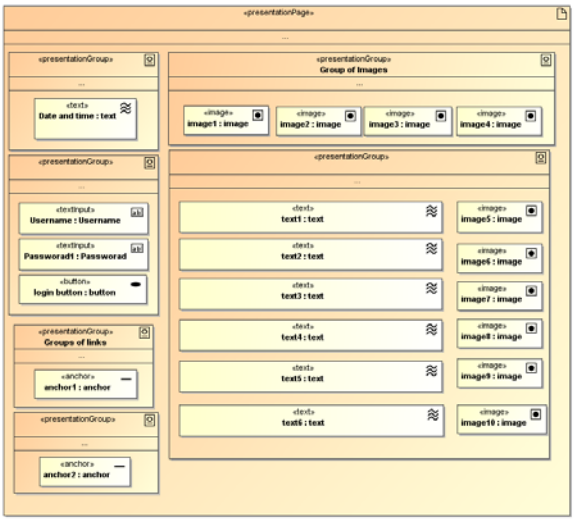

Figure 15. UML Presentation Model for Goalzz Homepage.

Also, we will work on handling Dynamic HTML pages (DHTML). By DHTML pages we mean pages which change its layout on the client side. So you will have a unique URL with different page layout segmentation. For example consider a faculty member website, page simply list the faculty member publication, if the user clicks on any publication the abstract of that publication will be shown on the page without the need to connect back to the server. Handling DHTML pages introduces two issues that need to be considered in our future work. First, we need to make sure that the segmentation process takes this into consideration. Second, we need to check if UWE notation is flexible and rich enough to capture and model such a dynamic behavior of the web page.

Acknowledgment

This research was supported in part by German Research Foundation (DFG), Higher Council for Science and Technology (HCST) and Jordan University of Science and Technology (JUST).

References

[1] G. A. D. Lucca, A. R. Fasolino, and P. Tramontana, "Reverse engineering web applications: the WARE approach," Journal of Software Maintenance and Evolution: Research and Practice, v.16, n.1-2, p.71-101, January-April 2004 “doi:10.1002/smr.281”.

© 2012 ACADEMY PUBLISHER

[2] E. J. Chikofsky and J. H. II Cross, "Reverse engineering and design recovery: a taxonomy," Software, IEEE, vol.7, no.1, pp.13-17, January 1990 “doi:10.1109/52.43044”.

[3] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, VIPS: a vision-based page segmentation algorithm, Microsoft Technical Report, MSR-TR-2003-79, November 2003.

[4] S. Tilley and S. Huang, "Evaluating the reverse engineering capabilities of Web tools for understanding site content and structure: a case study," Proceedings of the 23rd International Conference on Software Engineering, pp. 514-523, May 2001 “doi: 10.1109/ICSE.2001.919126”.

[5] S. Chung and Y.-S. Lee, "Reverse software engineering with UML for Web site maintenance," Proceedings of the First International Conference on Web Information Systems Engineering, vol.2, pp.157-161, 2000 “doi: 10.1109/WISE.2000.882874”.

[6] R. Virgilio and R. Torlone, "A Structured Approach to Data Reverse Engineering of Web Applications," Proceedings of the 9th International Conference on Web Engineering, San Sebastián, Spain, June 2009 “ doi:10.1007/978-3-642-02818-2_7”.

[7] B. Djelloul, M. Mimoun, and B. S. Mohamed, "Ontology based Web Application Reverse-Engineering Approach," INFOCOMP Journal of Computer Science, vol. 6, pp. 37-46, March 2007.

[8] C. Kroiß and N. Koch, "The UWE Metamodel and Profile – User Guide and Reference", Technical Report 0802, Ludwig-Maximilians-Universität München, p. 35, February 2008.

Natheer Khasawneh is an Assistant Professor in the Department of Software Engineering at Jordan University of Science and Technology since 2005. He received his BS in Electrical Engineering from Jordan University of Science and Technology in 1999. He received his Master and PhD degrees in Computer Science and Computer Engineering from University Akron, Akron, Ohio, USA in the years 2002 and 2005 respectively. His current research interest is data mining, biomedical signals analysis, software engineering and web engineering.

Oduy Samarah is a Master student in the Department of Computer Engineering at Jordan University of Science and Technology since 2009. He received his BS in Computer Engineering from Jordan University of Science and Technology in 2009. His current research interest is in Wireless sensor network, software engineering and web engineering.

Safwan Al-Omari is an assistant professor in the department of Software Engineering at Jordan University of Science and Technology. Dr. Safwan Al-Omari received his PhD degree in Computer Science from Wayne State University in 2009. He received his Master degree in Computer and Information Science from the University of Michigan-Dearborn and Bachelor degree in Computer Science from the University of Jordan in 2003 and 1995, respectively. His current research is in software engineering, service-oriented computing, and cloud computing..

Stefan Conrad is a Professor in the Department of Computer Science at Heinrich-Heine-University Duesseldorf, Germany. He was an Associate Professor in the Department of Computer Science at Ludwig-Maximilians- University in Munich, Germany, from 2000 to 2002. From 1994 to 2000, he was an Assistant Professor at the University of Magdeburg where he finished his ‘Habilitation’ in 1997 with a thesis on federated database systems. From 1998 to 1999, he was also a Visiting Professor at the University of Linz, Austria. He received his PhD in Computer Science in 1994 at Technical University of Braunschweig, Germany. His current research interests include database integration, knowledge discovery in databases, and information retrieval. He is a (co)author of two books (in German) and a large number of research papers.

© 2012 ACADEMY PUBLISHER