



SCHOOL OF
COMPUTER SCIENCE

Information
Management Group

HCW— WIMWAT Technical Report 3, July 2009

Exploring Widget Identification Techniques for Web Accessibility

A Review of the Literature

Alex Q. Chen and Simon Harper

Human Centred Web Lab
School of Computer Science
University of Manchester
UK

This study investigates the research surrounding Web accessibility issues, and the required knowledge and techniques to discover Web widgets from the Web page's source code. It provides a literature review of the coverage, and the overlap of previous studies for the Widget Identification and Modification for Web 2.0 Access Technologies (WIMWAT) project. Making the Web accessible is difficult, and at the rapid rate it is evolving, the Web needs additional help. Very few research have been done to identify Web widgets from the Web page's source code, and none has tried to evaluate the accessibility of the content manipulated by them. To fill this gap, we suggest widgets can be discovered from source code through reverse engineering, and evaluating the accessibility of the widget's manipulated content, so that patterns where inaccessible content may be produced can be modified into an accessible form.

HCW

Human Centred Web

WIMWAT

The aim of the Widget Identification and Modification for Web 2.0 Access Technologies (WIMWAT) project is to comprehend Web page's source code, so that Web widgets that produce inaccessible content can be identified, in order to modify them into an accessible form during development. The WIMWAT Web ages may be found at <http://hgw.cs.manchester.ac.uk/research/>.

WIMWAT Reports

This report is in the series of HCW WIMWAT technical reports. Other reports in this series may be found in our data repository, at <http://hgw-eprints.cs.man.ac.uk/view/subjects/wimwat.html>. Reports from other Human Centred Web projects are also available at <http://hgw-eprints.cs.manchester.ac.uk/>.

Contents

1	Introduction	1
1.1	Synopsis	2
2	Web Accessibility	2
2.1	Guidelines & Recommendations	2
2.1.1	Web Content Accessibility Guidelines (WCAG)	3
2.1.2	Evaluation and Report Language (EARL)	3
2.1.3	Authoring Tool Accessibility Guidelines (ATAG)	3
2.1.4	Accessibility Evaluation and Repair Tools (AERT)	4
2.2	Conformability	6
2.3	Assistive Technologies	7
2.4	Classifying Quality of Accessibility	10
3	Web Design	12
3.1	Designing for Usability	12
3.2	Maintainability & Organisation	13
3.2.1	Design Patterns	13
3.2.2	Pattern Libraries	14
3.2.3	Modeling Languages	16
4	Reverse Engineering	17
4.1	Purposes for Identifying Web Widgets	17
4.2	Coding Format	18
4.3	Code Comprehension	21
4.3.1	Matrices	23
4.3.2	Conceptual Models	25
4.3.3	Vectors	28
4.3.4	Searching for Instances	29
4.3.5	Annotation	29
5	Conclusion	30

Human Centred Web Lab
School of Computer Science
University of Manchester
Kilburn Building
Oxford Road
Manchester
M13 9PL
UK

Corresponding author:
Alex Qiang Chen
tel: +44 (161) 275 7821
chenqa@cs.man.ac.uk
<http://homepages.cs.manchester.ac.uk/~chenqa>

1 Introduction

The web is a medium that provides an environment where files are interlinked, and can be accessed publicly via the Internet. It is a heterogeneous combination of technologies, documents, recommendations, and guidelines. Since the proposal of the Web 2.0 concepts in 2004 [68], the Web has flourished with Rich Internet Applications (RIA). A lot of RIAs do not present content using the conventional ‘Page Model’ that most developers, and users are used to [64]. Instead they source and remix data from multiple sources, and present them in small chunks called micro-content. Hence, rather than reloading the entire page, information is updated dynamically in micro-content.

To update micro-content dynamically, this can be done using a few methods, namely either preloading the content, then using client-side scripting language to manipulate the presented content, or using remote scripting. Described in [26], one of the leading technologies of Web 2.0 is asynchronous JavaScript and XML (AJAX). This technology uses remote scripting to allow user-agents to communicate with the Web server, so that content is obtained from the Web server, and displayed in micro-content format, without requiring the Web page to reload.

Much of the Web data such as flight schedules, and all kinds of materials stored in databases, or embedded in the Web document is invisible. It will require Application Programming Interface (API) or widgets to extract this information [24]; this data is known as the ‘Deep Web’ [9]. Thus, the accessibility of this content relies heavily on whether these Web applications, and widgets are developed, so that they produce content in an accessible form. Discussed by Borodin et al., screen readers often faced problems when attempting to cope with the evolution of Web technologies [11]. The lag behind the new technologies makes it difficult, or sometimes even impossible for visually impaired users to access the information [15, 24]. Both Web content authors/developers, and user agents must support guidelines and specifications such as the Accessible Rich Internet Applications (ARIA) suite in order for users to benefit from the advancement, however, in practice, this coordination is often not achievable.

The Web evolve at an expeditious rate due to its popularity. This rapid change, and little economical benefits makes accessibility guidelines difficult to obey [73, 50, 80]. Another important factor is much of the Web content is created by non-professionals, and they lack the knowledge of the importance of accessibility.

In this report, we present the literature review for the WIMWAT project. This project aims to identify and modify possible Web widgets that produces inaccessible content during development. To do this work, a broad understanding of areas such as code comprehension, so that patterns of a Web widget can be recognised, and techniques used to distinguish the different widgets are required. After detecting the widget, we must understand the process of the widget, so that the accessibility of the content produced by the widget can be evaluated, and repaired if it is not in an accessible form. Thus, existing techniques used to evaluate Web accessibility are also being reviewed.

1.1 Synopsis

This report is structured as follows:

Section 2: Web Accessibility discusses the related guidelines and recommendations, research done to improve the conformability of the guidelines, and advancement of assistive technologies to aid people with disabilities.

Section 3: Web Design covers the components and considerations taken by a Website's designer, developer, and management when developing a Website, and the resources available to aid the process.

Section 4: Reverse Engineering describes the different techniques of detecting a Web widget pattern from the Web page's source code.

Section 5: Classifying Quality of Accessibility explores existing methods and tools used to classify whether the source code is accessible.

Section 6: Conclusion discusses the conclusions on the techniques and resources reviewed.

2 Web Accessibility

The practice to make Web pages accessible to all users, especially to those with disability refers to Web accessibility [89]. Harper and Yesilada described the Web accessibility field as a mechanism to provide equal opportunity to everyone [51]. Their review on Web accessibility and guidelines suggested that disabled people still face difficulties when accessing the Web.

To provide true Web accessibility, a number of guidelines were developed to provide recommendations for Web developers/authors, user-agents, authoring tools, and assistive technologies with an international standard to follow. However, very few developers are willing to rework old content so that they conform to the guidelines, mainly because there are a lot of new content that need to be created, and the benefits of Web accessibility affects only a small population of the Web users [80]. Furthermore, much of the Web content are developed by non-professionals [73], but the Web accessibility domain requires more professionalisation [24].

To encourage more to adapt their Website to be accessible, Richards and Hanson attempted to highlight the benefits of making Websites accessible so that they will be accessible by more people, and discusses methods that cost developers lesser to adopt to these changes [80]. In their discussions, they demonstrated that accessibility not only benefits people with disabilities, but also older adults, and suggested how semi-automatic tools can help the adaptation process for Websites to conform to the Web accessibility guidelines.

2.1 Guidelines & Recommendations

Section 508 and many other Web accessibility guidelines provide a method to ensure the accessibility of Web content. However, the World Wide Web Consortium

(W3C), Web Accessibility Initiative (WAI) [84] guidelines are commonly referred internationally as the standard to follow. Some of the related guidelines and recommendations by the W3C will be covered in the following.

2.1.1 Web Content Accessibility Guidelines (WCAG)

The WCAG developed by WAI layout a series of guidelines for Web content developers and authors, and Web accessibility evaluation tool developers. WCAG 2.0 [18] has taken over from WCAG 1.0 [22] since December 2008 for the advancement of technologies, ease to understand, and suggests a more precise automated testing and human evaluation. WCAG 2.0 suggest techniques for general and technology-specific such as WAI-ARIA [25], HTML/XHTML, CSS, and scripting.

Three levels are used to provide an indication of how well a Web page conforms to WCAG. This is based on four principles: perceivable, operable, understandable, and robustness that oversees the twelve guidelines suggested in WCAG 2.0 [18]. The conformance levels are A, AA, and AAA, with AAA being the most difficult to satisfy. The three levels are described as follow,

- **Level A:** the Web page satisfies all the Level A Success Criteria, or a conforming alternate version is provided.
- **Level AA:** the Web page satisfies all the Level A and Level AA Success Criteria, or a Level AA conforming alternate version is provided.
- **Level AAA:** the Web page satisfies all the Level A, Level AA and Level AAA Success Criteria, or a Level AAA conforming alternate version is provided.

The advancement of Web technologies introduced by the Web 2.0 concept brought up new problems, such as dynamic content updating. WCAG 2.0 attempts to provide an answer to this issues by including WAI-ARIA. A study by Thiessen and Chen investigates AJAX live regions using a chat example to illustrate the robustness of WAI-ARIA [91]. They have report that developing accessible Rich Internet Applications (RIA) is possible through WAI-ARIA. From this investigation it gives an indication of how WCAG will perform when obeyed.

2.1.2 Evaluation and Report Language (EARL)

The EARL provides a standard language for Web accessibility evaluation tools to communicate their results with other tools that wants to use them. It is a machine-readable format for Web accessibility evaluation tools, validators, and other types of content checkers to express test results [2]. The EARL documents are developed by the Evaluation and Repair Tools Working Group (ERT WG), under the umbrella of the W3C's WAI.

2.1.3 Authoring Tool Accessibility Guidelines (ATAG)

ATAG is another set of guidelines developed by the W3C's WAI. It describes to developers of authoring tools, how can their tools assist Web developers to produce accessible Web content, so that the content conforms to WCAG [54]. These

tools include those that transform documents into Web formats, and content layout management e.g., CSS formatting tools.

2.1.4 Accessibility Evaluation and Repair Tools (AERT)

AERT suggests techniques for Web accessibility validation tools so that HTML documents can be evaluated to conform to WCAG. Since AERT has been a W3C working draft since 26 April 2000 [81], this working draft only conforms to WCAG 1.0 [22]. The AERT document consists of fourteen guidelines that builds on WCAG 1.0, describing techniques for evaluation and repair tools, so that problems with accessibility can be addressed and repaired.

Currently a number of evaluation tools are available for Web developers, notably WAVE, Cynthia Says, RAVEn, Tidy, Total Validator, and Web Accessibility Inspector all provide industrial standard for evaluating Web accessibility.

WAVE

WAVE is a free online Web accessibility evaluation tool developed, and maintained by WebAIM [99], to help Web developers make their Web content more accessible. Instead of providing a complex technical report of the Web page it has evaluated, WAVE checks the Web pages or content submitted to it, for the compliance issues with many of the Section 508 and WCAG guidelines. Then it shows the areas of the original Web pages using icons and indicators, to reveal the accessibility of it. However, it does not check all the issues in these guidelines,

HiSoftware Cynthia Says

The HiSoftware Cynthia Says is a solution for Web content accessibility validation. It is developed and maintained by HiSoftware Inc., to identify problems in Web pages content submitted to it based on Section 508, and WCAG 1.0 guidelines [56]. HiSoftware Cynthia Says tool comes in three versions: the enterprise, desktop, and free Web service. The free Web service version of the tool is meant for educational purposes, and restrict its users to submit only one page from a site per minute.

RAVEn

The IBM Rule-based Accessibility Validation Environment (RAVEN) inspects and validates Java rich-client, and Web based Graphical User Interface (GUI) oriented applications for accessibility [57]. This tool is now part of the Eclipse-based¹, validation component in ACTF.

HTML Tidy

HTML tidy is a free utility developed by Raggett, and now maintained by the open source community at Source Forge. It has been developed to improve Web accessibility and usability of hypertext documents. Not only will it assist developers

¹<http://www.eclipse.org>

to tidy hard to read hypertext documents, it can also identify the areas where developers should concentrate, so that their Web pages will be more accessible to people with disabilities [77].

Total Validator

The Total Validator is an all in one solution that validates the hypertext markup language, and evaluates the Web page's accessibility. It validates the Web page against W3C Markup Specifications for Document Type Definition (DTD) HTML 2.0, 3.2, 4.0, 4.01, and XHTML 1.0, 1.1 [93]. The accessibility evaluator provided examines the Web page against the WCAG 1.0 and 2.0, and Section 508 standard. This tool also comes in a desktop tool and a Firefox² extension for fast, one click solution.

Web Accessibility Inspector

The Web Accessibility Inspector (WI) is a free program developed by Fujitsu³, for both Windows and Mac OS X operating systems [42]. Like most evaluators or validators, this software will highlight the parts that require modifications. This software provides the ability to evaluate Web accessibility to not only English literate users, but Chinese and Korean literate users too. The importance of this type of tool has an growing importance in the Web community. As investigated by O'Neil et al. in [67], as the Web gains its popularity, the distribution of non-English content Website is also growing. However, the WI was developed to evaluate Websites against the WAI WCAG 1.0 guidelines only. Hence, this tool will not be able to fully examine Web pages created using the Web 2.0 concepts.

Other Web Accessibility Evaluation Investigations

The Web Accessibility Initiative (WAI) guidelines are widely accepted as the international standards for Web accessibility such as WCAG [84]. However, WAI guidelines are constantly undergoing changes, to keep up to pace with the technologies; from WCAG 1.0 in 1999 [22] to WCAG 2.0 in 2008 [18]. This constant updates add on to the difficulty that many designers are already facing when trying to design Web pages that comply with these guidelines.

A study by Abascal et al. in 2004 discuss about a repair tool called EvalIris, developed to automatically evaluate Website's accessibility using sets of guidelines, and the guidelines can be easily updated or replaced [1]. This tool was created to check the Web page's accessibility based on WCAG 1.0 guidelines. The EvalIris was evaluated using the W3C's Techniques For Accessibility Evaluation and Repair Tools [81] working draft. It was reported that this tool successfully fulfill the objectives of the study, and demonstrated the importance of having an evaluating tool that is capable of accepting new sets of guidelines. Since the EvalIris system is an on going project, further enhancement were also proposed, so that a crawler can be integrated, an user interface was also suggested to this Web service, and a XML

²<http://www.mozilla.com/firefox/>

³<http://www.fujitsu.com>

translation help to assist the translation for new guidelines into the XML schema used by EvalIris [1].

2.2 Conformability

One of the most difficult thing to do when designing Web pages is to make them visually attractive, enhance their usability, and meet the accessibility guidelines [88]. The process to designing Web pages so that they meet the expectations, and guidelines during the development stage is conformability. The purpose of achieving conformability is to gain high viewership, by making the Web pages accessible to more.

To assist Web developers to design their Websites to conform to accessibility guidelines, beside producing guidelines such as Section 508 or WCAG, a number of Web accessibility evaluation tools such as [99, 57, 56] are also available. Although none of these tools can fully evaluate the Web accessibility guidelines on a Web page, they do provide useful directions, and a learning environment for Web designer and developers.

Making a Website accessible is difficult, and due to the fierce competition between major Websites, Web developers often overlook Web accessibility as it does not have much economical benefits [88, 80, 1]. Unless developers, authoring tools, and user agents comply to the guidelines and recommendations available, for example those developed by WAI [84], otherwise true accessibility cannot be achieved. Richards and Hanson suggested that research surrounding Web accessibility compliance should move beyond developing guidelines and validators. They suggested providing semi-automatic repair tools such as [88] or even doing the transformation on the fly [80].

Petrie and et. al. suggested a roadmap for accessibility research, and discussed about tools for automatic and manual evaluation for analysing Web page's accessibility [71]. The study also suggested that accessibility should included as a iterative process during development. However, this study is based on WCAG 1.0 guidelines, and it was highlighted that this set of guidelines do not include accessibility for dynamic micro content updates.

In a recent report to propose a joint study programme on accessible Web design by Hengstberger and et. al., highlighted the need for education at the academic level for experts in accessible Web design [53]. It was reported that universities confined their Web accessibility courses to either the post-graduate level, or restricted to some chapters in computer science. The study discussed about the aspects for developing a distance learning module, in the field of accessible Web design, at a European level. It lay out the components to be included in the programme, how to go about with the accreditation, and the types of authoring tools available. Educating professionals about Web accessibility can prepare the further generations for producing better accessible Websites, however, it is not a solution to control it.

Other attempts were also made to provide a designing tool for non-professionals, such as iWeb⁴, so that accessibility can be integrated into the non-professional design world. Since these tools are developed after the creation of accessibility guidelines, these guidelines could be incorporated into these tools from the ground up [73].

⁴<http://www.apple.com/ilife/iweb/>

However, accessibility guidelines are difficult, and this means that even with these tools, fully accessible Websites cannot be achieved.

Besides studies to transform inaccessible Web pages into accessible ones, and evaluation tools, framework like the Accessibility Tools Framework (ACTF), provides an extensible framework and commendable accessibility tools [37]. Using this framework, ACTF allows the creation and use of various type of accessibility tools, and integrate the exemplary ones into a single tooling environment on top of the Eclipse⁵ framework. Then by collaborating the different tools with each other, a comprehensive development environment can be created.

2.3 Assistive Technologies

Assistive technology provides an avenue for people who are unable to use conventional technologies such as the mouse, the screen/monitor, and the keyboard, to access Web content [38]. The Web was developed mainly for the visual medium, hence, this causes real problems for people with visual disabilities. Using the model depicted in figure 1, assistive technologies enable the Web to be accessible by more people. An example of assistive technology is a screen reader, this utility dictates the content presented on the screen to its users, thus, allowing people with visual disabilities to access the Web.

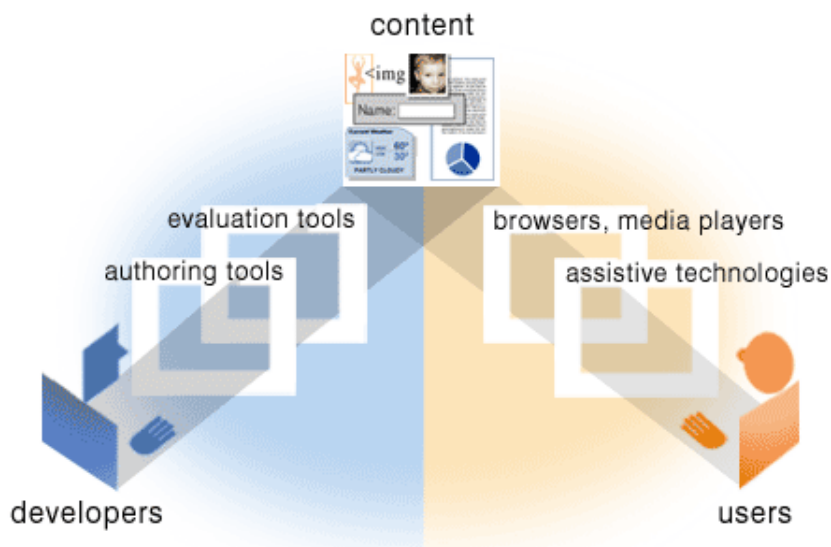


Figure 1: The different components required to develop a Web page, and accessing it. Figure courtesy of <http://www.w3.org/WAI/intro/relate.png>

The Accessibility Internet Browser for Multimedia (aiBrowser) was developed by Miyashita and et. al. to address the problems faced by screen readers when dealing with multimedia content, RIAs, and Flash [66]. Users of the aiBrowser can control the audio from the embedded media directly using short cut keys. A simple

⁵<http://www.eclipse.org>

alternative user interface using external metadata can be generated by the aibrowser for screen readers, this feature also be applied to dynamic content such as Flash. A diagram in figure 2 describes how the processes the content, and interacts with the users, and the Web server. An evaluation was conducted by comparing the aiBrowser with Job Access With Speech (JAWS)⁶ for the efficiency of the method used for navigation. From this evaluation, it is reported that even JAWS with Flash supports was having issues when trying to access Flash content. On the contrary, the aiBrowser is able to access Flash content by using its multimedia audio control features. By introducing the alternative user interface using external metadata, on general, lesser key strokes are required by the aiBrowser for both the novice or advanced operations.

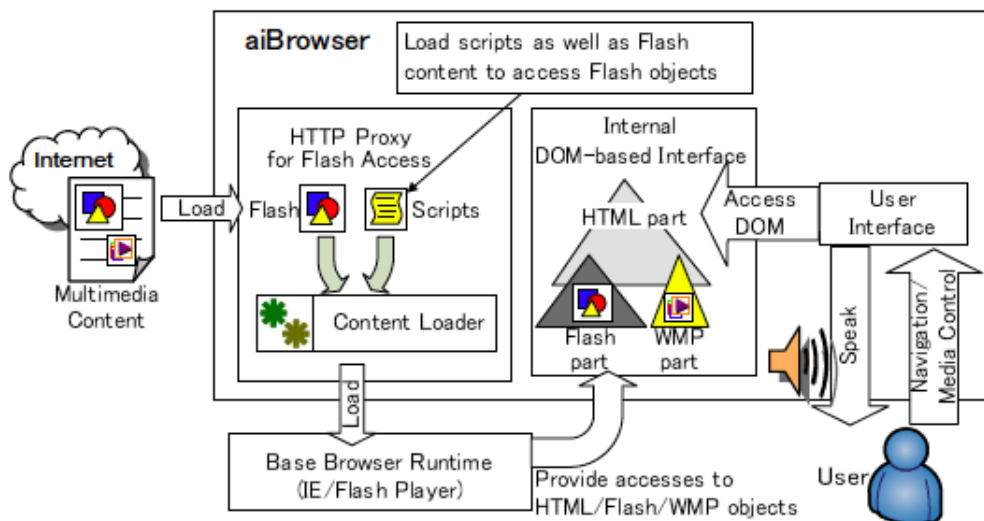


Figure 2: Overview of aiBrowser [66]

RIAs has allured Web developers to use technologies like AJAX and Flash to add interactivity to their Websites. Discussed by Borodin and et. al., the gap between the technologies and assistive technologies is widening, and they introduced an approach to unify the handling of micro content changes [11]. In this paper, they introduces Dynamo; an approach that handles dynamic micro content changes in Web page. To implement Dynamo, it is embedded within the framework of a non-visual Web browser called Hearsay. Using this approach, it os reported faster than existing interfaces when accessing dynamic content, and the users seems to prefer their approach.

CSurf is a context driven non-visual Web browser developed by Mahmud, Borodin, and Ramakrishnan to address the information overload problem faced by non-visual Web access [63]. It's objective is to provide a solution for people with visual disabilities to locate relevant information on the Web page quickly. This non-visual Web

⁶<http://www.freedomscientific.com/jaws-hq.asp>

browser uses two main algorithm to provide contextual browsing; Context identification and Relevant Block Identification. The architecture of the context driven non-visual Web browser, CSurf is depicted in figure 3. To evaluate the performance of CSurf, an experiment was conducted on thirty graduate students and three blind people, trained to use the CSurf Web browser, and they used only the keyboard and headphones to interact with it. The blind people were allowed to browse Websites that they are familiar, while the students used twenty five Websites for the evaluation. The reported results from the evaluation showed that using the context-driven approach saves time when browsing, and improve the browsing experience for people with visual disabilities.

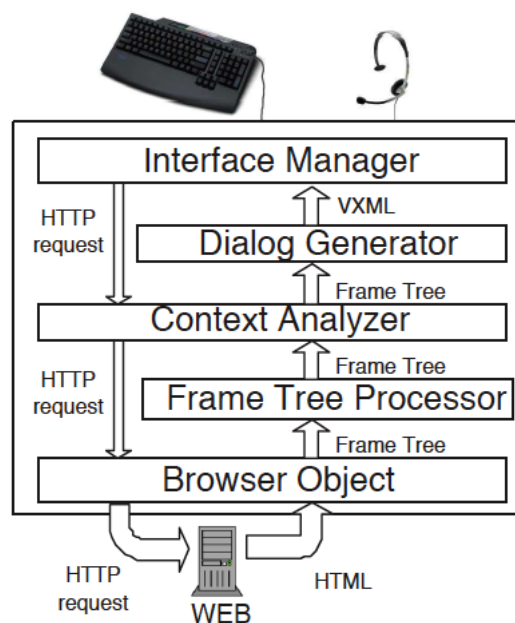


Figure 3: Architecture of CSurf, the context driven non-visual Web browser [63].

The advantages of providing a technology on the client's end such as the aiBrowser, will allow users with visual impairment to access content that uses interactive technologies. However, this type of technologies rely on developers to create Web pages that conforms to accessibility guidelines.

Interventions proposing solutions for a range of disabilities was also attempted, in particular, a software architecture proposed by Gonzalez-Pisano and et. al. [45]. The proposed system separates the functionality of a Website and its interface during runtime, and customises some interface elements according to the characteristics of the individual. The users information about their navigation process is updated continuously and stored during runtime. They believe that by integrating such a system with Web browsers, a large number of accessibility barriers for people with varying degrees of disability will be overcome, Websites usability will be improved, and adaptation to devices used for navigation. A prototype was successfully evalu-

ated during this study as a plugin for Mozilla Firefox Web browser. This prototype is specific for people with low visuality, but it is fully scaleable. This attempt by Gonzalez-Pisano and et. al. is a bold attempt to provide non-specific solution for people with disabilities. It also illustrated the importance, and possibilities of well written accessible code.

2.4 Classifying Quality of Accessibility

One of the processes in WIMWAT requires some form of evaluator to classify if the content produced by the widget is accessible. This suggested a review of different techniques for measuring accessibility. Several techniques and tools can be employed to classify Web accessibility. A good understanding of some of these measuring techniques is essential, when attempting to classify whether a widget produces accessible content.

Web accessibility is commonly measured by the conformance level of a Website using a set of guidelines. However, other measuring methods are also suggested by some, such as Web Accessibility Barriers (WAB) metrics [70, 49], Web Accessibility Quantitative Metric (WAQM) [96], and many more. Each of these methods provide Web developers with different solutions to measure a Website's accessibility from a different point of view.

Previously, Sullivan and Matson examined the content accessibility compliancy, and usability for the Web's most popular fifty Websites based on WCAG 1.0 [86]. They tested for the accessibility of the Websites' content by using a simple ratio shown in equation 1. This method compares the potential failures points, and the actual failure points.

$$FailureRate = \frac{ActualFailure}{PotentialFailure} \quad (1)$$

To determine if there was any interrelationship between accessibility and usability, a correlation between both the accessibility and usability results was analysed. The results from the usability tests were obtained from the LIFT tool, that was used to assess the fifty Websites selected in [86], while accessibility results was measured using equation 1. They reported that the results demonstrated some form of interrelationship, but further investigations are required to confirm this.

A metric to measure Web content accessibility based on 25 WCAG checkpoints is proposed by Parmanto and Zeng in [70]. This method is the WAB that quantitatively measure the Web content accessibility based on equation 2. The Websites with fewer accessibility barriers will have lower score, while a higher score means more barriers.

$$WABscore = \frac{\sum_p \sum_v (\frac{n_v}{N_v})(w_v)}{N_p}, \quad (2)$$

where p refers to the total number of pages on the Website, v denotes the total violations of the Web page, n_v is the actual number of violations, N_v indicates the number of potential violations, w_v denotes the weight of the violations in inverse proportion to WCAG priority level, and N_p is the total Web pages checked.

Parmanto and Zeng evaluated their novel approach proposed in [70] with a more complex method; a C5.0 machine learning algorithm [76]. The evaluation is divided into two main parts, the first part, examines 8 to 893 pages from 29 scientific journals for the accessibility of these journals, and a study of how Web accessibility performed over time was also done for this part. To evaluate how Web accessibility performed over time, the Food and Drug Administration (FDA) website was analysed between 1997 and 2004. The second part, evaluates the validity of the WAB score against the WCAG 1.0 level; non-rated, A, AA, and AAA. Results from the evaluation demonstrated that WAB produced results comparable with C5.0 machine learning algorithm, and meets the requirements as a measurement for scientific research.

More recently, Vigo et al. proposed a quantitative measurement for Web accessibility called Web Accessibility Quantitative Metric (WAQM) [96, 97], to accurately measure the accessibility of Web content. This approach can provide a means for quantifying quality assurance, information retrieval, and monitoring Web accessibility. They highlighted that the methods suggested by Sullivan and Matson in [86] was a native approximation, as it does not consider many factors such as the error impact, and the nature of the error. While Parmanto and Zeng suggestion to used WAB [70] do not allow its users to assess the accessibility for a single Web page. Vigo et al. in [96] worked out the WAB score for a single Web page is

$$WABscore = \sum \frac{ActualErrors}{PotentialErrors \times Priority} \quad (3)$$

The WAQM calculates the evaluation reports in XML, provided by the EvalAccess evaluation tool. It assess accessibility by providing a more precise measurement scale, rather than whether it is either accepted or rejected. This approach inherits the good points from other methods [86, 70], and automates the process. It assumes that all Web pages in a Website has equal importance, but this is not true in practice. Thus, in [97], Vigo et al. extended their approach, so that Web pages that are deeper in the hierarchy will have lesser impact on the final value.

Freire et al. did an evaluation for different Web accessibility metrics based on a set of important attributes [41]; set of guidelines used by the metric, coupling level of the metric with guidelines, type of evaluation (automatic/manual), whether the metric considers the site complexity, type of barriers weights coefficients, default values for barriers weights coefficients, checks if the metric 's report is comparable with any evaluations involving real users, the type of automated tool available, and whether the metric is use in large scale evaluations. From the evaluation, Freire et al. reported that WAB and WAQM couples very well with the WCAG guidelines, and WAQM was designed to work well with automatic evaluation tools. Evaluation such as [41] provide a quick and important investigation to help review the different types of methods, and to search for methods to assist our proposed study.

A method to measure Web accessibility is by estimating the severity of barriers faced by users when using the Website. Brajnik suggested a method called manual method for measuring barriers of accessibility (MAMBO) in [14]. An accessibility barrier is defined by Brajnik as "any condition that makes it difficult for people to achieve a goal when using the web site through specified assistive technology". MAMBO is a metric based on a probabilistic schema, that uses information gath-

ered from the barrier walkthrough accessibility reports. The barrier walkthrough accessibility reports contain estimations of the severity of the barriers, the function of the impact, and its frequency. The method relies on the level of experience of the evaluator in accessibility of Web technologies, accessibility evaluations, and the knowledge of assistive technologies. Thus, MAMBO is limited by its ability to cope with accessibility barriers missed out by evaluator, and he suggested further investigations surrounding this issue should be pursued.

3 Web Design

When designing a good Website, it involves a few key components, these are the appearance, usability, maintainability, and organisation. Figure 4 illustrates the iterative process visited by the management when designing and developing, or maintaining a Web application. Understanding the sequence and components required when designing a Website, will indirectly give a clearer concept of how a specific widget is formed when attempting to reverse engineer it. The key components that are generic for designing most Websites will be discussed.

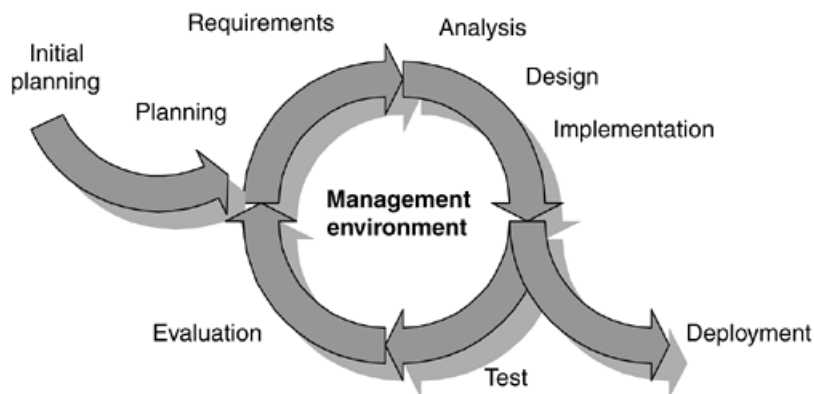


Figure 4: The iterative process to develop and maintain an application. [60]

O'Reilly describes the different and evolving design models in software development, and business over the internet in [68]. He highlighted that the Web 2.0 is a fuller realization of the true potential of the web and how it can help to achieve richer user experiences. With the Web 2.0 concepts, software is no longer delivered as a product, but a service. Thus, ending the release cycle of software. It encourages programs to be lightweight, so that they will be good for reusability.

3.1 Designing for Usability

A usable Website allows their users to access the Web page, and find their way around the Website quickly and efficiently. This means Web pages must be accessible to all, including people with disability.

Discussed by Maurer, designing Web pages with RIAs that are usable and accessible is difficult, and she goes on to elaborate the key challenges that designers will face when ensuring their applications are usable [64]. However, accessibility problems with RIAs seem to be an increasing issue as the Web evolves with the Web 2.0 concepts [48]. Gwardak and Pålshorp then explored the usage of guidelines when designing Web pages with RIAs. From the investigations, they suggested a set of guidelines that is a hybrid between the desktop user interface, and the Web Content Accessibility Guidelines (WCAG) 1.0.

In order to help assistive technologies to adapt to Web 2.0 technologies, studies such as Borodin et al. [11], and Brown and Jay [16] were conducted for audio browsing.

3.2 Maintainability & Organisation

The life cycle of a Web page goes through a number of stages, and some of these stages loop round endlessly during the life span of the Website. Hence, good organisation and structure to the Website will be required to ensure the quality, efficiency, and ease of maintaining it.

During the designing phase of the Website, keeping an unambiguous record of the Web pages designed is an essential key to provide a good documentation. Fortunately, there are useful tools such as Unified Modeling Language (UML) [23, 59], Web Modeling Language (WebML) [20], and Hypertext Design Model (HDM) [44] to provide a means of communicating difficult to explain concepts of the Web applications to the designers, developers, and users [31]. These documentations will prove to be useful during maintenance when the developers, or new developers, need to revisit the problem again.

Highlighted by some studies, the maintenance task of a Website is not easy, and often requires tools such as [29, 31, 85] to help ease this work. This is because quite often, due to the competitiveness, advancement of the Web, and pressure imposed by their competitors, documentations are lacking, and there is little time for the developers to turn the Web applications around.

3.2.1 Design Patterns

When developing a Web page, different components come together to form the user interface, and the engine that controls its behavior. Commonly, to describe the different parts of an application, design patterns are used. Design patterns were described by Christopher Alexander in 1979, as “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice” [3]. He illustrated why one pattern differs from another, and how it is reliant on the surroundings and the people that created it. This issue relates to the granularity issues of the design patterns, and varied interpretation by [43, 17, 4, 98, 61].

Gamma et al., also known as the Gang of Four (GoF), described design patterns as reusable elements for object-oriented software approach in [43]. The book

catalogued 23 design patterns that is widely referred by many in this field [65, 35, 85, 52, 94]. It gives a full description for each pattern, including the names and intention of the pattern. Additional design patterns for interaction are suggested in [4], so that users can control the visibility of information. Gamma et al. emphasised the importance of determining the object granularity, and it can vary tremendously from one person to another, and the implementation of the object. This is the key to make a flexible design, and ensure the design patterns can adapt to changes later in its life cycle. A closer look at patterns reveals that some patterns provides the structure of a application into subsystems, and others support the refinements of the subsystems and components [17].

The documentation of design patterns will evolve over time, thus incorporating flexibility into the design is vital. Commonly the design patterns are not properly documented, and it may result to violation of the constraints and properties of the design patterns. Furthermore, Bosch discussed about the issues with breaking down the design patterns, and reorganising it to suit the structure of an application [13]. This could lead to traceability problems in the later part of the application life cycle. Investigations by Dong, Zhao, and Sun aimed to tackle the evolution problems face when adding or removing design elements from existing design patterns during the transformation process [36]. They proposed an automated process for the evolution of design patterns. Such a system can reduce errors and missing parts, however, if the pattern overlaps other pattern instances in the design, errors and inconsistencies may arise from it. This study highlighted the importance to design our methodology for identifying Web widgets, so that it will be able to handle the evolution of widget's patterns.

Benatallah et al. suggested four additional patterns in 2002 for architecting, and managing composite Web services [8]. They proposed these patterns because they believe "Web services are loosely coupled Internet-accessible software entities delivering functionalities provided by business applications and processes". These patterns provides a solutions to the recursive problems to integrate and provide Web services. However, the Web has evolved since, and with Web 2.0 concepts, and many new business models, the definitions of these patterns need to be revisited.

Described by Mikkonen, formalising design patterns will allow rigorous reasoning, and also supports a software engineering view for the development. He explains how to formalised temporal behaviours of design patterns to create the specifications for complex systems [65]. According to Mikkonen, formalisation of a pattern can remove ambiguity, so that temporal behaviours of patterns can be rigorously attended to. This technique can be useful to ensure the soundness of the design concept when designing a system, however, it does not bridge a bidirectional relationship between the designing and implementation stage.

3.2.2 Pattern Libraries

There are numerous design patterns libraries available both online and printed on hardcopy. The libraries that are closely related to our work when defining our widget's definition are

Yahoo! Developer Network's Design Pattern Library

The Yahoo! Developer Network's design pattern library⁷ is one of the most referred pattern library online. It provides a range of Applications Programming Interface (API) and their documentations that forms the Yahoo! User Interface (YUI) library. These resources are maintained by an active community working on the YUI, that constantly updates their resources and pattern libraries.

Welie's Patterns in Interaction Design

The Welie's Patterns in Interaction Design⁸ is a similar pattern library with Yahoo! Developer Network's design pattern library, except it does documentations solely for patterns only. A few of the patterns overlaps those in Yahoo!, however, the descriptions may vary slightly. This provides a good comparison when choosing the correct pattern. A certain point, due to the different opinions on the patterns, this library's granularity resulted to a large set of patterns for some pattern category.

Wikipatterns

Wikipatterns⁹ is a set of tools for everyone to share patterns that have helped them [100]. In this library, patterns are grouped into four categories; People Patterns, People Anti-Patterns, Adoption Patterns, and Adoption Anti-Patterns. Wikipatterns describe anti-patterns as patterns that represent a negative behaviour or consequence. These are situations where developers do not want to occur, but they are common.

UI Pattern Factory

The UI Pattern Factory¹⁰ includes a range of user interface design pattern library, mixed with user interface gallery. Similar to both Welie's Patterns in Interaction Design and Yahoo! Developer Network's design pattern library, this library has its own set of definitions and naming for the patterns. Again due to the difference in experience, and granularity of the patterns, the patterns varies in a different point of view.

The Design of Sites

The book [95], provides a referential material for the principles, patterns, methodologies, and best practices for creating Websites. It describes the patterns in a similar fashion as Yahoo! Developer Network's Design Pattern Library, however, they categorised the patterns according to their usage, and they do not provide custom APIs. Being a hardcopy library, this means that it will provide a consistent source for referential, and it do not change when the pattern evolve.

Each of the pattern libraries discussed gives a unique perspective to a pattern or set of patterns. Some of these libraries' patterns can be employed to assist the conceptualisation, and justification of a pattern. The collection of patterns in Yahoo!

⁷<http://developer.yahoo.com/ypatterns/>

⁸<http://www.welie.com/patterns/>

⁹<http://www.wikipatterns.com/display/wikipatterns/Wikipatterns/>

¹⁰<http://uipatternfactory.com/>

Developer Network’s Design Pattern Library, and Welie’s Patterns in Interaction Design are the most closely related set of patterns for our proposed work. Thus, these attributes make them the best candidates for our primary source.

3.2.3 Modeling Languages

Due to the heterogeneous combination of technologies, guidelines, and recommendations, developers have found it useful to use modeling languages during development. Although Conallen explained how to employ UML for building Web applications [23], some such as [7, 59] proposed extending UML to model the Web applications, and [44, 20] proposed a new form of modeling language specifically for the Web.

Garzotto, Paolini, and Schwabe, proposed a model based approach called Hypertext Design Model (HDM) [44] that is not take up by many. It is a modeling mechanism to describe concisely about a hypertext application. The HDM allows the authors to conceptualise the application without getting into the implementation details, and it can provide a means to communicate between the designers, developers, and users.

Koch and Kraus describe a systemic design methodology using UML in [59]. The UML-based Web Engineering (UWE) methodology provides a systematic and stepwise construction of models, for personalized and semi-automatic generation. Koch and Kraus chose UML because its a known semantics of these diagrams, however, due to its limitations, it do not support fine grained component concepts, and it lacks the capability of integrating different modeling techniques. UWE is a methodology that provides the guidelines for a systemic, and stepwise approach to construct models for Web applications. To attain semi-automatic generation of Web applications from design models, the ArgoUML tool is implement to support the construction of UWE design models. Refinements for modeling dynamic aspects of Web application is foreseeable as future work, and Koch and Kraus aims to develop a tool for systematic design modeling and XML publishing framework for semi-automatic generate Web applications. To cope with systematic building, an extension for the ArgoUML case tool is being developed.

Baresi et al. introduced the W2000 framework for designing Web applications in [7]. This framework integrates UML and HDM [44], extending the expressiveness of UML. Baresi et al. demonstrated that unlike “traditional” hypermedia, Web applications allow their users to navigate, activate operations and transactions, they are dynamic, and evolvable.

Web Modeling Language (WebML) was proposed by Ceri, Fraternali and Bongio in [20] to allow Website designers to express the core features of a Website at a high level, without getting involved with the detailed architectural details. An example of how WebML can be used illustrate the composition, and navigation specification is demonstrated in figure 5.

The Extensible Markup Language (XML) syntax is support by WebML, and can be fed to software generators to produce the Websites. A Website can be specified in WebML with four models; structural, Hypertext, Presentation, and Personalisation model. The Structural Model expresses the relevant entities and relationships between the data content of the Website. The Hypertext Model, describes the com-

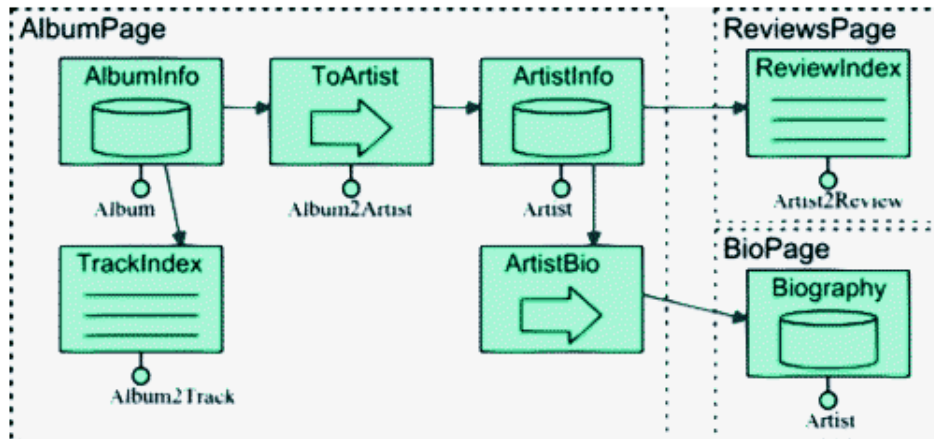


Figure 5: Example of WebML used for describing the composition and navigation specification. [20]

position of the Web pages, and how they are interlinked to form the hypertext. The Presentation Model illustrates the appearance of the Web pages, and lastly, the Personalisation Model allows designers to model the users and user groups in the structure schema. WebML is a high-level specification language for designing data-intensive Web applications, and it builds on several proposals for Web design language and hypermedia. This language is the backbone of Toriisoft, an environment for designing Websites at its advanced development state [20].

4 Reverse Engineering

The process to reverse engineer a Web widget is an analytical study of the widget's source code, so that the technological principles can be discovered. Through reverse engineering, design information such as the decisions made earlier by developers, can help to understand how the widget was developed. An illustration of the flow of processes for forward engineering, and reverse engineering is presented in figure 6. The purpose for reverse engineering a widget will allow the production of content by the widget to be evaluated, so that inaccessible content produced can be modified. The proposed work requires the different reverse engineering techniques for its analysis. To understand the depth of the investigation, selecting the correct coding format to conduct the evaluation, being aware of the types of coding formats and their different version when attempting to comprehend it, and understanding the different techniques available for the task is crucial.

4.1 Purposes for Identifying Web Widgets

The techniques used to identify Web widgets are employed for many reasons. Guha et al. detects AJAX Web applications (widgets) so that the possibility of AJAX intrusion detection can be investigated [47]. However, some attempt to identify

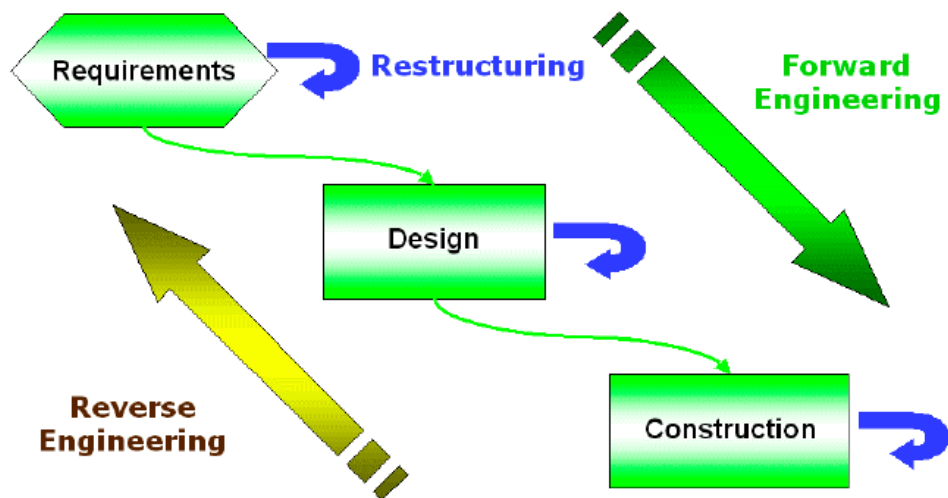


Figure 6: Reverse and forward engineering [92]

Web applications to make maintenance of the Website easier [29, 36].

Before WCAG 2.0 became a set of accessibility guidelines, Di Lucca et al. attempted to identify and rectify accessibility issues in a client's page code [30]. They used conceptual models to model the Web accessibility guidelines (WCAG 1.0 and WCAG 2.0), then they proposed a accessibility model to identify Web accessibility problems on a Web page. A case study was conducted to determine the conformance of ten Websites to evaluate the concept. Twenty Web pages from each of the ten Websites were examined. The results detailed that most of the Websites except two were found not to have Web accessibility problems. This study reconfirms the need and demand for a rectification tool on the developer's end, however, it only examines superficial code, and it does not attempt to understand the semantics behind the scripting code and the hypertext code, and it also does not relate the different objects that forms the Web application.

Without understanding these elements, the multifarious Web applications that extensively reuses objects that are created within a code, and APIs, it will not be possible to modify the Web applications, so that they are in an accessible form. Furthermore, the Web has evolved to be much more dynamic since then, and WCAG 2.0 has now become a guideline.

4.2 Coding Format

The Web allows a combination of various Web document formats to form together to make the Website work. These formats include hypertext and styling languages to client-side scripting languages. Understanding the different languages, and their different versions is vital when comprehending them. A wide selection of programming languages are also available to evaluate our concept. However, selecting a suitable language is important to avoid bottlenecks that the language may impose.

Hypertext Markup Language (HTML)

The HTML DTD has evolved into a number of versions together with evolution of the Web. From a recent study conducted on Web Evolution, Chen presented that the popularity of the latest version of DTDs tends to be preferred and adopted by Web developers [21]. Currently as reported, the Web pages mainly use the DTD of HTML 4.01, Extensible Hypertext Markup Language (XHTML) 1.0 and 1.1. This is because XHTML 2.0 [6] and HTML 5 [55] are both currently still in their working draft state.

HTML 5 attempts to be backwards compatible, however, suggested by [101], backward compatibility of different versions of HTML/XHTML should be dealt by the user-agent. Making HTML 5 backward compatible may make it more confusing for the users, and adds complexity to the validators and work surrounding Web accessibility. Another proposal to make HTML 5 an object-model approach rather than the element format, will foreseeably add complexity when machines attempt to comprehend hypertext documents.

Cascading Style Sheets (CSS)

CSS is a language to include style to Web documents, and controls the appearance of text, i.e., the size, colour, font family, etc., the Web page's layout. It comes in a number of levels and profiles to deal with different devices; namely, level 1, 2 and 3, mobile profile 1.0 etc [12]. CSS allows the document structure to be separated from the presentation, so that developers/designers can have precise control of the Web page's layout and styling without interfering with the markup content [58]. Through this method, Web content authors can concentrate on the content itself, and the accessibility aspect of it.

JavaScript

JavaScript is a client-side scripting language to enhance the functionality of Web browsers. It is an interpreted programming language that has object-oriented capabilities, embedded in most popular Web browsers. Although JavaScript is not Java, the syntactic core of the language resembles C, C++, and Java [39]. This client-side scripting language does it by manipulating the Web browser's functionality from the behavior layer. However, the language may perform slightly differently when used in different Web browsers. Patents such as [27] attempt to create a JavaScript client framework, so that it will provide object-oriented features, and enable cross-window and cross-frames communications. This framework wants to standardise the scripting language, so that it will be independent from the different types of browsers, and the different versions.

Web widgets often use JavaScript to assist them to manipulate the presentation of Web content, and the content itself. Web pages that employ the Web 2.0 concept may even use this scripting language to conduct remote scripting, so that they can communicate, and retrieve additional information from the Web server without reloading the Web page.

Due to the popularity of this client-side scripting language [21], our proposed work should initial focus on widgets developed this type of client-side scripting language to prove our concept.

PHP

PHP: Hypertext Preprocessor (PHP) is a general-purpose server-side scripting language that can be embedded into hypertext documents [72]. It is a imperative and object-oriented scripting language that is distributed as a free software under the PHP License (this is not the same as the GNU General Public License). However, the PHP engine only allocates 8MB of run time memory for each page. Although this issue can be altered by changing `memory_limit` value on the server's `php.ini` file, this may still impose a bottleneck to our methods. This is because one of the foreseeable problem is, when search for some instance, it may require to temporary store nested multiple dimensional arrays in arrays when using any of the metric techniques (see section ??) to comprehend large Web page's source code. These methods will quickly fill the allocated memory. PHP was created as a server-side scripting language that focuses on rapid development of Web services. This is a lightweight model that is reusable by multiple Web pages. Thus, this scripting language may lack some of the capability to conduct the more intensive, computational task for reverse engineering work.

Java

Java is a high-level standalone or server-side programming language that can run virtually across multiple platforms [87]. Much of the syntax in Java derives from C and C++, but only simpler. Java provides a wide range of APIs and utilities for rapid development, and a Java Virtual Machine (JVM) is required to execute the Java application [26]. The memory management of this language is automatically done by the garbage collector. Although the programmer has the control to deallocate memory, it do not prevent memory leaks to occur, because Java allows both the program, and the garbage collector to manually, and automatically manipulate it. When this occurs, it can be difficult to debug the problem.

Java provides its own set of regular expressions utility by using the standard `java.util.regex` package [46]. The syntax is very similar to Perl, and the range of APIs provided by this utility is comprehensive enough to conduct most text manipulation, or pattern searching.

Python

Python is a dynamic, interpreted programming language that can be quickly coded, and is comparable to Perl, Java or Ruby [74]. Both natural expression and procedural coding are included, and it is object-orientated. The regular expression module provides matching operations similar to Perl's regular expression, with slight variations [75]. However, when comparing this language with PHP or JAVA popularity, this language may be limited by the range of APIs available for us to quickly build an testing platform to evaluate our concept.

Perl

Perl is an interpreted, cross platform programming language that supports procedural, and object-oriented programming [69]. Due to its capability to manipulate text, and provide a rapid development cycle, Perl claims to be the most popular programming language for the Web [90]. It is an Open Source software, licensed under the GNU General Public License (GPL). Perl works with HTML, XML, and other mark-up languages, and its database integration interface (DBI) supports third-party databases including Oracle, MySQL, and others. This programming language is supported on Unix, Macintosh (OS 7-9 and X), Windows, and many more operating systems.

Ruby

Ruby is a open source object-oriented programming language that blends syntax of its creator's, Yukihiro "matz" Matsumoto favourite languages; Perl, Smalltalk, Eiffel, Ada, and Lisp [82]. Unlike many object-oriented languages, classes can include a module, and exploits all its methods for free. Exception handling features, like those provided in Java and Python, is also provide in Ruby for handling errors. Ruby was created for those who require a scripting language that is more powerful than Perl, and more object-oriented than Python. This programming language even comes with an open-source Web framework (Ruby on Rails) for rapid Web development [83].

4.3 Code Comprehension

Described by many [62, 79, 29] code/program comprehension plays a vital role in software evolution and software maintenance. The purposed work comprehends the source code to identify the regions in the source code that produce inaccessible content, so that it can be modified.

Rajlich and Wilde described in [79], the bottom-up theory for program comprehension is based on parts of code that the programmer recognises; called chunks. Commonly, smaller chunks are nested within a larger chunk of code. This is the similar concept of pattern granularity discussed in [3, 43]. However, Rajlich and Wilde described that most programmers tends to used the "as needed" strategy as programs become larger, and usually it is not possible to completely comprehend the source code due to the time available [79]. The provides a reason for comprehending the Web page's source code for instances of Web widgets because of the "as needed" strategy approach.

Documentation of an application has been commonly found lacking for both software applications [35], and Web applications [29]. Accessing the documentations for Web applications is even more difficult, because applications can reuse APIs from different sources, and sometimes, documentations for these APIs are not available [35].

Di Lucca et al. investigates an approach to recover user interaction design patterns for Web application, so that it will make the maintenance task of the Website easier [29]. They proposed a three phase process to search for patterns in Web

applications; the training phase, candidature phase, and a validation phase. The architecture used in the study for recovering Web interaction design patterns is depicted in figure 7.

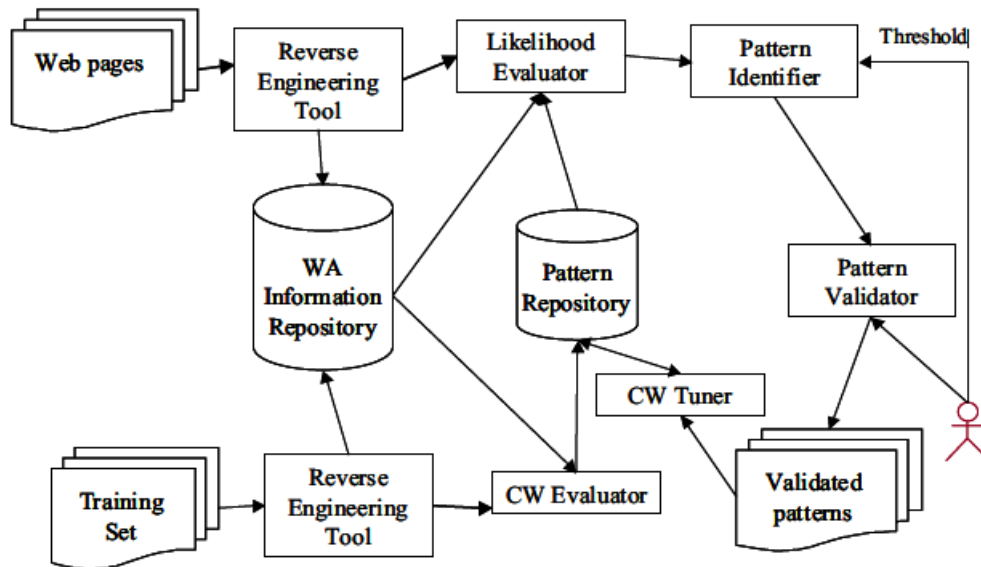


Figure 7: The architecture used in [29] for the Web Interaction Design Pattern Recovery System

Six Web interaction design patterns were chosen to evaluate the approach on a small set of training samples in [29]. These patterns include Guestbook, Login, Poll, Registration, Search, and Sitemap. The experiment obtained 79% precision. Then a wider set of 208 Web pages was conducted on another experiment for comparison using the same approach achieved 66% precision. This study demonstrated the diversity of the Web, however, the application of this study is different from ours. It does not examine the scripting code, semantics, and relationships between these elements. The results reported in [29] can provide us with a benchmark to base our methodology upon when attempting to identify the user interaction design pattern of the Web widgets.

In the early 1990s, studies such as [62] were conducted to assist reengineering and maintenance of old code, so that object-like features from a non object oriented language can be recovered, and the code can be transform into the object-oriented concept. This kind of study provides a useful referential source for general concepts when identifying objects from the source code, while applying it for a different purpose.

Liu and Wilde suggested an approach that uses a Global Based Object Finder and a Types Based object Finder to answer to the object oriented concept transformation [62]. The Global Based Object Finder searches for global and persistent data within the code, and creates the relationships of these data with the routines that manipulate them. While the Type Based Object Finder relies on the data types to

establish the relationships between the data types, and the routines that use them as a parameter or to return values. However, this proposal is a semi-automatic approach, and it requires human inputs.

These two studies gave an overview of code comprehension techniques used for investigating on Web applications [29], as well as to assist the transformation of one concept to another [62]. However, a review in [35], demonstrated a number of different techniques that can be employed to cater for different purpose. A comparison between the techniques was also presented. Based on this review and additional literatures, we grouped the different code comprehension techniques into five general categories that are presented in the follow sections.

4.3.1 Matrices

Matrices are used in the process when comprehending the source code. Dong, Lad, and Zhao proposed a novel approach to discover design patterns directly from the source code. They developed a tool called DP-Miner that uses matrix and weight to discover design patterns from the source code [32]. A pattern matrix is used to represent the system structure, where the columns and rows are all the classes in the system. In each cell, the value represents the relationships between the classes. A system matrix is used for the structure of each design pattern. Using this method, design patterns can be discovered by matching the two matrices; if they match, a instance of the pattern is found.

Three analysis were conducted to efficiently discover the instances of a design pattern. These analysis include a structural analysis of the system and design patterns to be discovered, a behavioral analysis to understand the dynamic relationships among the participating patterns, and a semantic analysis that searches for the design documentation of the source code, in-line comments in the source code, and clues from the naming convention of the classes. Finally the tool was tested on the Java.awt package in JDK 1.4 to search for instances of the Adapter pattern. The AWT package consists of 346 files, a total of 485 classes, and 111 interfaces. it is reported in [32] that it took 2.44 seconds to discover 21 Adapter pattern instances.

The approach proposed by Dong, Lad, and Zhao in [32] is a good proof-of-concept the existence of studies surrounding identifying patterns from the source code. However, this study focus on standalone or object-oriented programs, that do not cover the same magnitude of Web applications. Furthermore, after discovering the patterns, no further work was done to the discovered patterns.

A novel approach, DP-Miner, is based on matrix and weigh to discover instances of design pattern from object-oriented software systems [34]. Dong and Zhao developed this tool to provide an alternative approach, by searching for design pattern directly from the source code. This approach uses XMI-based intermediate representations for structural analysis, while the behavioral and semantic analysis is extracted from the system's source code directly, so that false positives can be reduced. The overall architecture of the approach is illustrated in figure 8. This work seems to be at its preliminary stages, and the authors have reported some inconsistency in their approach. The study only comprehend object-oriented software systems, and not on the level like the Web. It only attempts to discover the design

patterns from the source code, and it does not do anything to it like us. Discovering design patterns is not easy, but understanding the application of the code, and checking whether it is of an accessible form, so that modifications can be made to it is a much more challenging, and intensive problem.

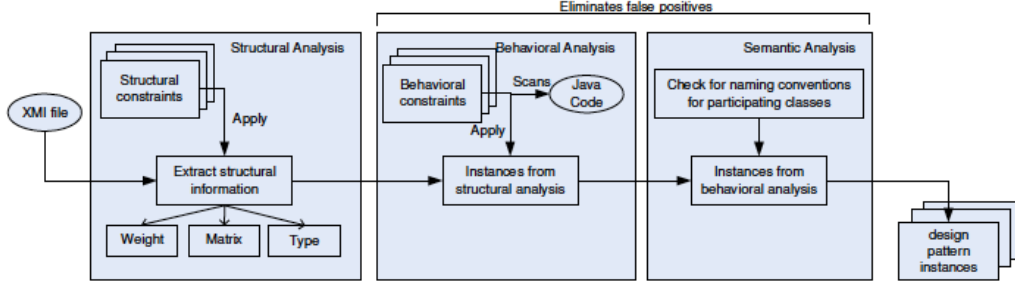


Figure 8: The overall architecture of the approach for [34]

In a recent study by Dong, Sun, and Zhao, an attempt to discover design patterns by using template matching [33]. The motivation behind this investigation is due to the poor design documentation. Template matching is done by matching a pattern matrix with a system matrix to determine whether a design pattern exist. This paper extends this approach by calculating the normalised cross correlation (CCn) between the pattern and system matrix, so that the degree of similarity between the design pattern and the part of a system can be computed.

$$CCn = \sum \frac{f(x) \cdot g(x)}{|f(x)| \cdot |g(x)|}, \quad (4)$$

where $f(x)$ and $g(x)$ are two vectors, and $x = 1, \dots, n$.

Eight design features were attempted to be encoded from four large open-source systems in [33]. The results presented are convincing, and they claimed that their approach can avoid false positive detection, that are caused by individual matches.

A study similar purpose to the study by Dong, Sun, and Zhao in [33], is [94] by Tsantalis et al.. They proposed a method that uses the similarity scoring between the graph vertices, instead of template matching. It calculates the similarity scores between the vertices of the system and the pattern graph. This allows patterns that are both in this basic form, and modified versions to be detected. Tsantalis et al. evaluated their approach with three popular open-source projects that employs design patterns extensively and systemically. It was reported that the results from the evaluation demonstrated the approach is precise and accurate, but a few false negatives surfaced, and no false positive was detected.

Studies by [32, 34, 33, 94] provide evidence that design patterns can be discovered from the source code, either by template matching, similarity scoring, or using matrix and weight. However, these investigations focused around standalone or legacy systems that do that have the same degree of complexity. Thus, some of the methods suggested can be reused but it has to be modified, or applied from a

lower level, so that it will work with the vase combinations of technologies that make the Web works.

4.3.2 Conceptual Models

Conceptual models allows its users to express the meaning of terms and concepts, and to find the correct relationships between the different concepts. it removes ambiguous terms and meanings, so that projects are more robust and reliable [40]. A common notation used to describe a conceptual model is the UML.

Modeling languages such as UML are often employed to provide a conceptual modeling of the system developed during the designing process. However, design patterns will evolve over time, and due to the lack of good documentations, problems arises from this process. Dong, Zhao, and Sun proposed an automated process that transforms the UML model of a design pattern application into its evolved form [36]. This method defines the evolutions of design patterns process using XML Metadata Interchange (XMI) in the primitive level and the pattern level, and the XMI format is then translated into XSLT transformation rules (see figure 9). A semantic Web checker based on the Java Theorem Prover (JTP) was also developed to ensure the evolution process consistency. Dong, Zhao, and Sun envisioned that automating the evolution process of design patterns can reduced errors, inconstancy, and missing parts for the evolved design patterns.

A number of studies were done on conceptual models to reverse engineer a Web application, and most of these studies surround the Ubiquitous Web Application (UWA) framework. This framework comprises a methodology and set of meta-models for user-centred designs for Web applications [10].

Bernardi et al. proposed a semi-automatic recovery of user-centred conceptual models from Web applications in [10]. This approach is defined according to the UWA design framework. The UWA Hyperbase model is a user-centred conceptual model that represents the Web application's content, organisation (entities and components), and semantic associations between the entities where navigation paths are derived. The process taken by Bernardi et al. to identify the UWA entities and semantic associations types is depicted in figure 10. Agreed by most Web applications design methods proposed, including [20, 10], three models can be used to define the development of Web applications; the Contents model, the Navigation model, and the Presentation model. To support the recovery process of the UWA conceptual models from a Web application, the architecture designed to carry out this task is described in figure 11.

A case study was conducted on the semi-automatic recovery of user-centred conceptual models to verify (1) the group of keywords extracted, (2) the candidate's UWA entities, (3) to ensure no UWA entity was undetected, (4) the semantic association of the candidate's UWA was identified, and (5) no UWA semantic association was undetected [10]. The case study was based on four Web applications. The results seems to represent that the authors' definition of a Web application is a Website in this paper, and reported result suggest that the group of keywords identified in HTML templates do not represent the application domain's concept. These results do not tell much, because a Website consist of a lot of small applications within

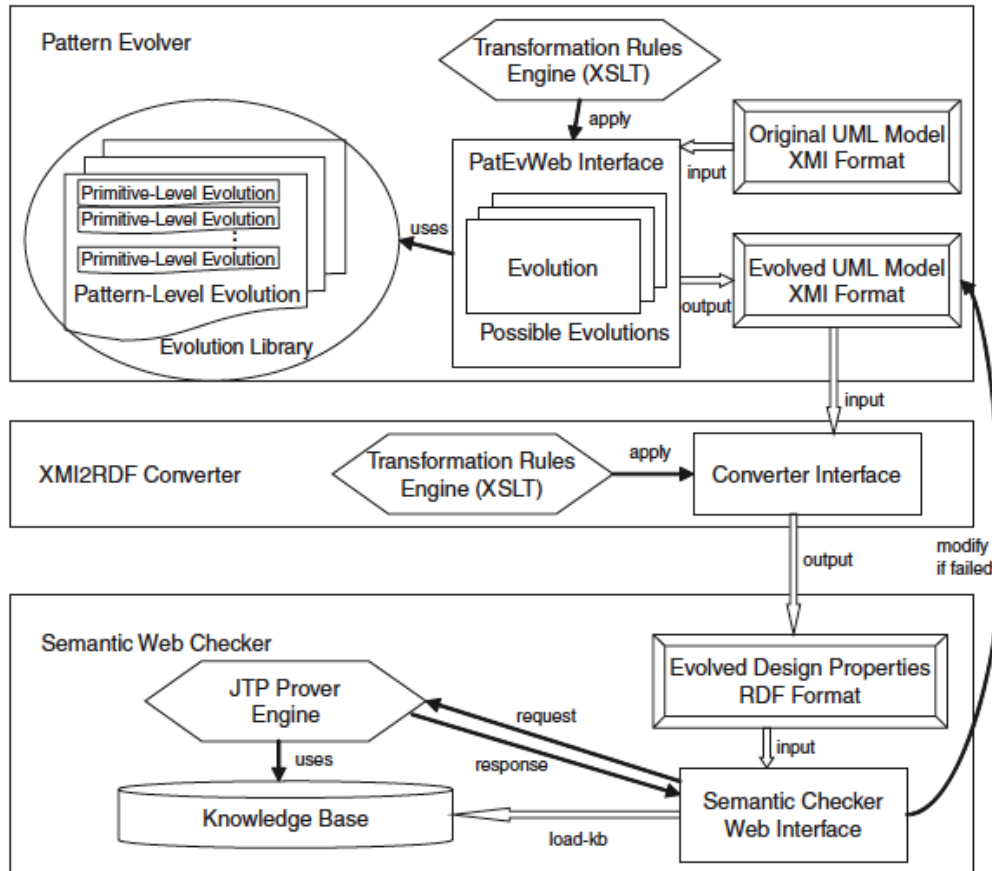


Figure 9: The overall architecture of the approach suggested in [36]

it. In fact, on a lot Web pages, it will consist of more than one small applications. The concept of a Web application in [10] seems to be a reuse of the conventional standalone applications concepts, and this is not the case for the Web medium. We believe by redefining this concept, it will produce a different set of results for the investigation.

Studies were conducted to reverse engineer the conceptual user-centred models from Web applications. Di Lucca, Distanto, and Bernardi proposed a reverse engineering conceptual model approach in [28], so that structural information and UWA models can be extracted from the Web application. To recover UWA models, static analysis process was conducted to Web applications to identify the entity types, semantic association types, collection types, and node types and cluster Types. A tool named RE-UWA was developed, and evaluated using three experiments. The first experiment consist of simple Web pages developed by undergraduate students, each having only one of the UWA concept implemented. The second experiment was conducted on a small size Web application developed by graduate students, that allows users to make prediction on football matches. Then the third experiment consist of

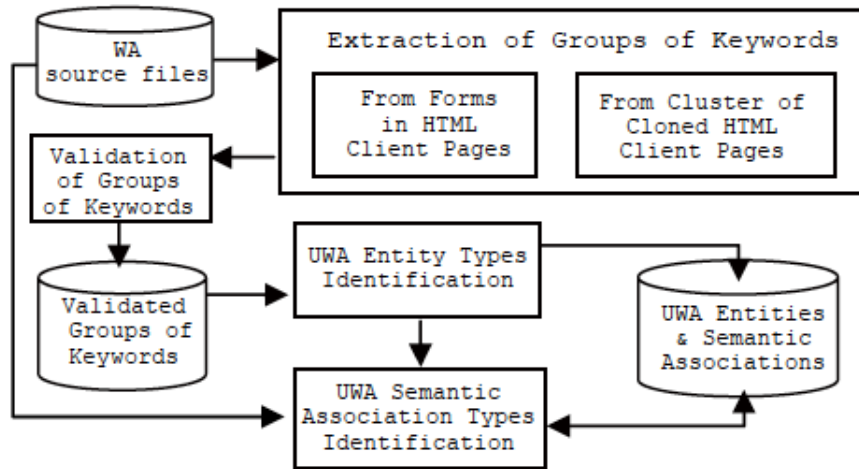


Figure 10: The identification process of UWA entities and semantic associations types in [10]

a large set of client pages downloaded. The first two experiments achieved precise identification of entities, collection, and associations, however, the third experiment showed a lower precision when identifying entities and associations. Di Lucca, Distantem, and Bernardi demonstrated that this approach is feasible and have good effectiveness, but refinements to the approach is expected to improve the approach's effectiveness and precision.

Distante, Parveen, and Tilley demonstrated in [31], a technique for reverse modeling, so that it will be able to comprehend Web application transaction design and implementation, with the absence of original program documentation. The study attempted to recover information from an existing Web application, and feed them as design input to an extended version of UWA transaction model. The main steps of the reverse modeling technique was reported to be inspected by a human, and the study was a demonstration of the group's concept. Thus, they suggested a automated tool as their future work for this approach.

Conceptual models such as [28], employs visual representation of a application, such as UML, and their associated automated environments to comprehend the source code claims to provide potential improvements, however, Hendrix et al. highlighted that these studies generally are limited in scope, and do not necessary designed to scale up for industrial practice [52]. The gang of them investigated into the effectiveness of visualisation representation when performing source code comprehension. The investigation compares the Control Structure Diagram (CSD) results with 2 groups of participants results, one involving senior-level class of software engineering students, and a few graduate students, while another group, made up of undergraduates students with little programming experience. It was reported that the CSD responded positively for shortening the response time, and have better correctness in the experiments. Thus, it was suggested by Hendrix et al. to

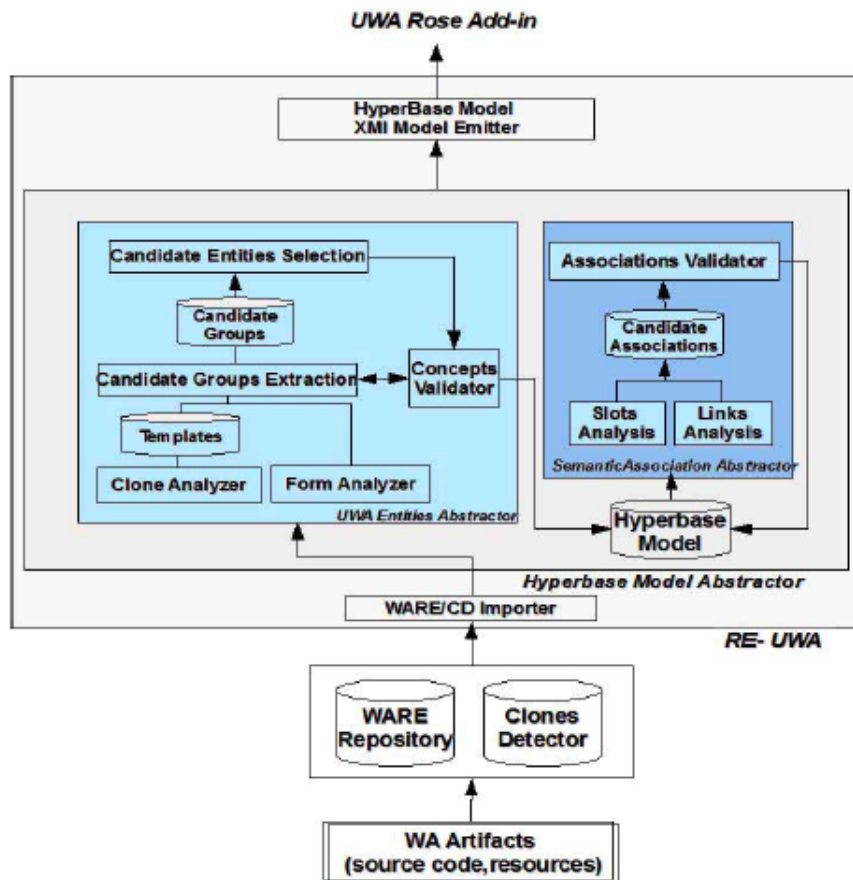


Figure 11: Architecture for the Hyperbase model abtractor module in [10] for the Re-UWA environment.

include the visualisation representation approach, as it can improve productivity and reliability.

4.3.3 Vectors

Vectors can represent a pattern structure or process of a source code, so that further analysis can be conducted to comprehend the code. Studies such as [94, 33] uses this method to model the relationships between classes for the matrices, so that the matrices can be manipulated easily, and it can clearly convey the matrix's concept to engineers and computer scientists.

An investigation by Canfora et al. to improve the techniques to identify object from the code, a statistical approach was introduced [19]. Although the objective of the study was to provide a solution to make software comprehension easier, but the concept from this approach can be deployed to assist code comprehension for Web applications. It was reported that a major problem in software comprehension is to comprehend the relationships between the system's components. The technique

suggested by Canfora et al. exploits a graphical method, and applied a statistical technique to the interconnections of a bipartite graph. The graph is plotted by using an iterative clustering algorithm, that terminates when the graph forms a candidate object. An object in this study is referred as a collection of data items and routines. From an experiment conducted to evaluate the quality of the proposed algorithm, demonstrated that it identifies the routines that are likely to introduce undesired connections, and the case studies demonstrated that some degree of human interaction is required.

4.3.4 Searching for Instances

A common method to search for a pattern when comprehending the source code is to look for instances of the pattern. Quite often, as seen in [94, 33], this method is combined with other techniques to further comprehend the source code.

Stencel and Wegrzynowicz proposed an automatic pattern recognition method, that can detect nonstandard implementations of design patterns [85]. The motivation for this proposal is because the existing approaches are not perfect, and sometimes fail to capture the source code intend. The aim of this proposal is to capture the intend of created patterns, as well as discover as many variants implementation method as possible. A tool was trained to recognise four design patterns, the Singleton, Factory Method, Abstract Factory, and Builder, and tested against two other state-of-the-art tools. The demo source of “Applied Java Patterns” (AJP), which provides an exemplary Java implementation of Gang of Four patterns, were used to test the three tools. The results proven that this method is flexible and efficient.

The techniques used in [85] proven a valuable approach for our purpose, however, this study was based on Java source code. Hence, the degree and complication of have a mixture of different languages in a similar application was not tested, and it does not attempt to examine the associations of the different patterns, and the purpose of having the pattern like our proposed work.

4.3.5 Annotation

Developers can annotate the application they are developer either directly with the source code, or separately as a documentation. Rajlich demonstrated the usefulness of having a separate incremental documentation for a software using the Web [78]. A software was developed as a hypertext notebook medium for programmers to record the understanding and observations about the software. Each component is annotated in different partitions of the documentations, starting from the root file. This form of annotation proves useful for the developers of the application, however, these documentations are not available for Websites, and documentation in the source code of the Web document will be useful.

Annotation can also be used as a technique to insert additional information such as semantics and notes in the Web page’s source code. Asakawa and Takagi developed an Accessibility Transcoding System, so that completed inaccessible Web pages can be converted into accessible Web pages [5]. Two components are used for the annotations; one for structural annotations, and the other for commentary annotations. The structural annotations are used to reorder the visually fragmented

groupings according to their importance, while the commentary annotations provides useful descriptions of the contents.

To overcome the fatigue of site-wide annotation authoring using the technique described by Asakawa and Takagi in [5], a new algorithm, “Dynamic Annotation Matching” was developed by Takagi et al. in [88], to automatically determine the appropriate annotations, and a authoring tool, “Site Pattern Analyzer” was also developed to support the annotator, visualizing the annotation matching status. An experiment was setup to test for the effectiveness of our algorithm and tools. USA Today’s Website was chosen for the experiment. It was reported in total 245 annotation files were created in 30 hours and 20 minutes, thus they succeeded to create annotation files for every target page without skipping any pages. These results demonstrated that annotation-based transcoding can improve Web Accessibility for visually impaired users, however, the time taken for annotation authoring is a bottleneck for this technique.

The approach used by Takagi et al. in [88] demonstrated the power of annotation, but it has to be used with great care to tackle the issues with this technique. This study was conducted in 2002, and during that time, most Web pages still uses the ‘Page Model’ concept described by [64]. Hence, it does not include Web pages with Web widgets, and dynamic micro content.

5 Conclusion

The literature review presented covers the required related work on Web accessibility, components of Web designing, existing techniques for reverse engineering, and methods to classify the quality of a Web page’s accessibility. We reviewed the available resources to aid our work in sections 3.2.2 and 4.2, and the different available tools in sections 3.2.3 and subsubsec: AERT. Studies such as [94, 36, 62, 19] provide us with useful techniques to carry out the code comprehension task, so that we can reverse engineer the Web widgets from the Web pages source code. The studies presented are either for standalone applications, or assumes that documentations are available, since they mainly focus for maintaining the Websites. This is may not true in practice, as documentations for Web APIs are not always available.

The reviews of research conducted to measure Web accessibility for a Website or Web page were discussed. Studies such as [86, 70, 96] equips us with the knowledge, so that we can effectively identify the accessibility problems produced by the widgets. With this knowledge, we can discover the accessibility issues produced by these widgets, and modifications can be done.

The studies presented do not cover the same degree or purpose of our propose work. From our review, some of these studies require to be updated due to the advancement of the Web, and none has tried to evaluate the accessibility of the content manipulated by Web widgets. This review provides us with useful knowledge, and acts as the basis of our work. From this literature review, we can conclude that the work WIMWAT has proposed to be investigated is vital to Web accessibility.

References

- [1] J. Abascal, M. Arrue, I. Fajardo, N. Garay, and J. Tomás. The use of guidelines to automatically verify web accessibility. *Universal Access in the Information Society*, 3(1):71–79, March 2004.
- [2] S. Abou-Zahra and S. L. Henry. Evaluation and Report Language (EARL) Overview. <http://www.w3.org/WAI/intro/earl.php>, March 2009.
- [3] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.
- [4] M. Arvola. Interaction design patterns for computers in sociable use. *Int. J. Comput. Appl. Technol.*, 25(2/3):128–139, 2006.
- [5] C. Asakawa and H. Takagi. Annotation-based transcoding for nonvisual web access. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 172–179, New York, NY, USA, 2000. ACM.
- [6] J. Axelsson, M. Birbeck, M. Dubinko, B. Epperson, M. Ishikawa, S. McCarron, A. Navarro, and S. Pemberton. XHTML 2.0. <http://www.w3.org/TR/xhtml12>, July 2006.
- [7] L. Baresi, F. Garzotto, and P. Paolini. Extending uml for modeling web applications. *Hawaii International Conference on System Sciences*, 3:3055+, 2001.
- [8] B. Benatallah, M. Dumas, M. C. Fauvet, F. A. Rabhi, and Q. Z. Sheng. Overview of some patterns for architecting and managing composite web services. *SIGecom Exch.*, 3(3):9–16, 2002.
- [9] M. K. Bergman. The deep web: Surfacing hidden value. *Digital Web Magazine*, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.363>, September 2001.
- [10] M. L. Bernardi, G. A. Di Lucca, and D. Distanto. Reverse engineering of web applications to abstract user-centered conceptual models. In *Web Site Evolution, 2008. WSE 2008. 10th International Symposium on*, pages 101–110, 2008.
- [11] Y. Borodin, J. P. Bigham, R. Raman, and I. V. Ramakrishnan. What's new?: making web page updates accessible. In *Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 145–152, New York, NY, USA, 2008. ACM.
- [12] B. Bos. Cascading Style Sheets. <http://www.w3.org/Style/CSS/>, August 2009.
- [13] J. Bosch. Design patterns as language constructs. *Journal of Object-Oriented Programming*, 11(2), 1998.

-
- [14] G. Brajnik. Measuring web accessibility by estimating severity of barriers. In *Web Information Systems Engineering WISE 2008 Workshops*, pages 112–121. Springer Berlin, 2008.
- [15] A. Brown and C. Jay. A review of assistive technologies: Can users access dynamically updating information? <http://hcw-eprints.cs.man.ac.uk/70/>, The University of Manchester, August 2008.
- [16] A. Brown and C. Jay. SASWAT Technical Requirements. <http://hcw-eprints.cs.man.ac.uk/79/>, The University of Manchester, October 2008.
- [17] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *A System of Patterns: Pattern - Oriented Software Architecture*. John Wiley & Sons, 1996.
- [18] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W. Chisholm, J. Slatin, and J. White. Web Content Accessibility Guidelines (WCAG) 2.0. <http://www.w3.org/TR/WCAG20>, December 2008.
- [19] G. Canfora, A. Cimitile, and M. Munro. An improved algorithm for identifying objects in code. *Software: Practice and Experience*, 26(1):25–48, 1996.
- [20] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, pages 137–157, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.
- [21] A. Q. Chen. Web Evolution. Master’s thesis, The University of Manchester, School of Computer Science, Kilburn Building, Oxford Road, Manchester, M13 9PL, UK, September 2008.
- [22] W. Chisholm, G. Vanderheiden, and I. Jacobs. Web Content Accessibility Guidelines 1.0. <http://www.w3.org/TR/WCAG10>, May 1999.
- [23] J. Conallen. *Building Web Applications With UML*. Addison-Wesley, 2nd edition, 2003.
- [24] M. Cooper. Accessibility of emerging rich web technologies: web 2.0 and the semantic web. In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 93–98, New York, NY, USA, 2007. ACM.
- [25] J. Craig, M. Cooper, L. Pappas, R. Schwerdtfeger, and L. Seeman. Accessible Rich Internet Applications (WAI-ARIA) 1.0. <http://www.w3.org/TR/wai-aria>, February 2009.
- [26] P. Deitel and H. Deitel. *JAVA: How to Program*. Pearson Education, Inc., 7th edition, 2007.

-
- [27] T. Dencker, C. Fischer, and A. Röessler. Javascript client framework. United States Patent, <http://www.google.co.uk/patents?id=c5WpAAAAEBAJ>, March 2008.
- [28] G. A. Di Lucca, D. Distanto, and M. L. Bernardi. Recovering conceptual models from web applications. In *SIGDOC '06: Proceedings of the 24th annual ACM international conference on Design of communication*, pages 113–120, New York, NY, USA, 2006. ACM.
- [29] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana. Recovering interaction design patterns in web applications. In *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on*, pages 366–374, 2005.
- [30] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana. Web site accessibility: identifying and fixing accessibility problems in client page code. In *Web Site Evolution, 2005. (WSE 2005). Seventh IEEE International Symposium on*, pages 71–78, 2005.
- [31] D. Distanto, T. Parveen, and S. Tilley. Towards a technique for reverse engineering web transactions from a user’s perspective. In *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, pages 142–150, 2004.
- [32] J. Dong, D. S. Lad, and Y. Zhao. Dp-miner: Design pattern discovery using matrix. In *Engineering of Computer-Based Systems, 2007. ECBS '07. 14th Annual IEEE International Conference and Workshops on the*, pages 371–380, 2007.
- [33] J. Dong, Y. Sun, and Y. Zhao. Design pattern detection by template matching. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 765–769, New York, NY, USA, 2008. ACM.
- [34] J. Dong and Y. Zhao. Experiments on design pattern discovery. In *PROMISE '07: Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, pages 12+, Washington, DC, USA, 2007. IEEE Computer Society.
- [35] J. Dong, Y. Zhao, and T. Peng. A review of design pattern mining techniques. *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, 2008.
- [36] J. Dong, Y. Zhao, and Y. Sun. XSLT-based evolutions and analyses of design patterns. *Software: Practice and Experience*, 39(8):773–805, 2009.
- [37] Eclipse. Accessibility Tools Framework (ACTF). <http://www.eclipse.org/proposals/actf/>, Accessed on 16 August 2009.
- [38] A. D. Edwards. Assistive technologies. In S. Harper and Y. Yesilada, editors, *Web Accessibility: A Foundation for Research*, pages 141–162. Springer, 2008.

-
- [39] D. Flanagan. *JavaScript: The Definitive Guide*. O'Reilly, 4th edition, 2002.
- [40] M. Fowler. *Analysis Patterns Reusable Object Models*. Addison Wesley, 1996.
- [41] A. P. Freire, R. P. M. Fortes, M. A. S. Turine, and D. M. B. Paiva. An evaluation of web accessibility metrics based on their attributes. In *SIGDOC '08: Proceedings of the 26th annual ACM international conference on Design of communication*, pages 73–80, New York, NY, USA, 2008. ACM.
- [42] FUJITSU. Web Accessibility Inspector. <http://www.fujitsu.com/global/accessibility/assistance/wi/>, Accessed on 20 August 2009.
- [43] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- [44] F. Garzotto, P. Paolini, and D. Schwabe. Hdm—a model-based approach to hypertext application design. *ACM Trans. Inf. Syst.*, 11(1):1–26, 1993.
- [45] J. Gonzalez-Pisano, M. Rodriguez-Fernandez, M. Gonzalez-Rodriguez, J. Bobes-Bascaran, and J. Garcia-Marsa. A system for dynamic adaptation of web interfaces based on user interaction requirements. In *Computers Helping People with Special Needs*, pages 276–283. Springer-Verlag, 2008.
- [46] J. Goyvaerts. Using regular expressions in java. <http://www.regular-expressions.info/java.html>, Accessed on 19 August 2009.
- [47] A. Guha, S. Krishnamurthi, and T. Jim. Using static analysis for ajax intrusion detection. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 561–570, New York, NY, USA, 2009. ACM.
- [48] L. Gwardak and L. Pahlstorp. Exploring usability guidelines for rich internet applications. Master's thesis, Department of Informatics, Lund University, June 2007.
- [49] S. Hackett, B. Parmanto, and X. Zeng. Accessibility of internet websites through time. *SIGACCESS Access. Comput.*, (77-78):32–39, 2004.
- [50] S. Harper. Web evolution and its importance for supporting research arguments in web accessibility. In D. De Roure and W. Hall, editors, *Proceedings of the First International Workshop on Understanding Web Evolution (WebEvolue2008): A prerequisite for Web Science*, pages 51–54, Southampton, UK, April 2008. Web Science Research Initiative.
- [51] S. Harper and Y. Yesilada. Web accessibility and guidelines. In S. Harper and Y. Yesilada, editors, *Web Accessibility: A Foundation for Research*, pages 61–78. Springer, 2008.
- [52] D. Hendrix, J. H. Cross, and S. Maghsoodloo. The effectiveness of control structure diagrams in source code comprehension activities. *Software Engineering, IEEE Transactions on*, 28(5):463–477, 2002.

-
- [53] B. Hengstberger, K. Miesenberger, M. Batusic, N. Chelbat, and A. Rodríguez García. Joint study programme on accessible web design. In *Computers Helping People with Special Needs*, pages 182–189. Springer-Verlag, 2008.
- [54] S. L. Henry and M. May. Authoring Tool Accessibility Guidelines (ATAG) Overview. <http://www.w3.org/WAI/intro/atag.php>, December 2008.
- [55] I. Hickson and D. Hyatt. HTML 5. <http://www.w3.org/TR/html5>, April 2009.
- [56] HiSoftware Inc. HiSoftware Cynthia Says. <http://www.contentquality.com>, Accessed on 19 August 2009.
- [57] IBM. RAVeN On Accessibility. <http://www-03.ibm.com/able/resources/raven.html>, Accessed on 19 August 2009.
- [58] I. Jacobs and J. Brewer. Accessibility Features of CSS. <http://www.w3.org/Style/CSS/>, August 1999.
- [59] N. Koch and A. Kraus. The expressive power of uml-based web engineering. <http://dx.doi.org/http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.7588>, 2002.
- [60] P. Kruchten. *The Rational Unified Process: An Introduction*, page 7. Addison-Wesley, Boston, MA, 2nd edition, 2000.
- [61] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall PTR, 1998.
- [62] S. S. Liu and N. Wilde. Identifying objects in a conventional procedural language: an example of data design recovery. In *Software Maintenance, 1990., Proceedings., Conference on*, pages 266–271, 1990.
- [63] J. U. Mahmud, Y. Borodin, and I. V. Ramakrishnan. Csurf: a context-driven non-visual web-browser. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 31–40, New York, NY, USA, 2007. ACM.
- [64] D. Maurer. Usability for rich internet applications. Digital Web Magazine, http://www.digital-web.com/articles/usability_for_rich_internet_applications/, February 2006.
- [65] T. Mikkonen. Formalizing design patterns. *Software Engineering, International Conference on*, 1998.
- [66] H. Miyashita, D. Sato, H. Takagi, and C. Asakawa. Aibrowser for multimedia: introducing multimedia content accessibility for visually impaired users. In *Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pages 91–98, New York, NY, USA, 2007. ACM.

-
- [67] E. T. O’Neill, B. F. Lavoie, and R. Bennett. Trends in the evolution of the public web (1998 - 2002). *D-Lib Magazine*, 9(4), April 2003.
- [68] T. O’reilly. What is web 2.0: Design patterns and business models for the next generation of software. *Social Science Research Network Working Paper Series*, 2007.
- [69] O’Reilly Media Inc. perl.com: Documentation. <http://www.perl.com/pub/q/documentation/>, Accessed on 28 August 2009.
- [70] B. Parmanto and X. Zeng. Metric for web accessibility evaluation. *Journal of the American Society for Information Science and Technology*, 56(13):1394–1404, 2005.
- [71] H. Petrie, C. Power, and G. Weber. Web accessibility - automatic/manual evaluation and authoring tools. In *Computers Helping People with Special Needs*, pages 334–337. Springer-Verlag, 2008.
- [72] PHP.net. PHP: Hypertext Preprocessor. <http://www.php.net/manual/en/faq.php>, Accessed on 22 August 2009.
- [73] C. Power and H. Petrie. Accessibility in non-professional web authoring tools: a missed web 2.0 opportunity? In *W4A ’07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 116–119, New York, NY, USA, 2007. ACM.
- [74] Python community. About python. <http://www.python.org/about/>, Accessed on 19 August 2009.
- [75] Python community. Python library reference: Regular expression operations. <http://www.python.org/doc/2.5/lib/module-re.html>, Accessed on 23 August 2009.
- [76] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [77] D. Raggett. Clean up your Web pages with HTML TIDY. <http://www.w3.org/People/Raggett/tidy/>, Accessed on 19 August 2009.
- [78] V. Rajlich. Incremental redocumentation using the web. *Software, IEEE*, 17(5):102–106, 2000.
- [79] V. Rajlich and N. Wilde. The role of concepts in program comprehension. In *Program Comprehension, 2002. Proceedings. 10th International Workshop on*, volume 0, pages 271–278, Los Alamitos, CA, USA, 2002. IEEE Computer Society.
- [80] J. T. Richards and V. L. Hanson. Web accessibility: a broader view. In *WWW ’04: Proceedings of the 13th international conference on World Wide Web*, pages 72–79, New York, NY, USA, 2004. ACM.

-
- [81] C. Ridpath and W. Chisholm. Techniques for accessibility evaluation and repair tools. <http://www.w3.org/TR/AERT/>, April 2000.
- [82] Ruby community. About ruby. <http://www.ruby-lang.org/en/about/>, Accessed on 30 August 2009.
- [83] RubyOnRails.org. Rails framework documentation. <http://api.rubyonrails.org>, Accessed on 30 August 2009.
- [84] Shawn. Web Accessibility Initiative. <http://www.w3.org/WAI/>, August 2009.
- [85] K. Stencel and P. Wegrzynowicz. Detection of diverse design pattern variants. In *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*, pages 25–32, 2008.
- [86] T. Sullivan and R. Matson. Barriers to use: usability and content accessibility on the web’s most popular sites. In *CUU '00: Proceedings on the 2000 conference on Universal Usability*, pages 139–144, New York, NY, USA, 2000. ACM.
- [87] Sun Microsystems. Learn about java technology. <http://www.java.com/en/about/>, Accessed on 19 August 2009.
- [88] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Site-wide annotation: reconstructing existing pages to be accessible. In *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies*, pages 81–88, New York, NY, USA, 2002. ACM.
- [89] J. Thatcher, C. Waddell, S. Henry, S. Swierenga, M. Urban, M. Burks, and P. Bohman. *Constructing Accessible Web Sites*. Apress, July 2003.
- [90] The Perl Foundation. The perl directory: About perl. <http://www.perl.org/about.html>, Accessed on 28 August 2009.
- [91] P. Thiessen and C. Chen. Ajax live regions: chat as a case example. In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 7–14, New York, NY, USA, 2007. ACM.
- [92] S. Tilley and S. Huang. Evaluating the reverse engineering capabilities of web tools for understanding site content and structure: a case study. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 514–523, Washington, DC, USA, 2001. IEEE Computer Society.
- [93] Total Validator. Total Validator. <http://www.totalvalidator.com>, Accessed on 20 August 2009.
- [94] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis. Design pattern detection using similarity scoring. *Software Engineering, IEEE Transactions on*, 32(11):896–909, November 2006.

-
- [95] D. K. van Duyne, J. A. Landay, and J. I. Hong. *The Design of Sites: Patterns for Creating Winning Web Sites*. Prentice Hall PTR, 2nd edition, 2006.
- [96] M. Vigo, M. Arrue, G. Brajnik, R. Lomuscio, and J. Abascal. Quantitative metrics for measuring web accessibility. In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 99–107, New York, NY, USA, 2007. ACM.
- [97] M. Vigo, G. Brajnik, M. Arrue, and J. Abascal. Tool independence for the web accessibility quantitative metric. *Disability and Rehabilitation: Assistive Technology*, 4(4):248–263, 2009.
- [98] J. M. Vlissides, J. O. Coplien, and N. L. Kerth. *Pattern Languages of Program Design 2*. Addison-Wesley Publishing Company, 1996.
- [99] WebAIM. WAVE. <http://wave.webaim.org>, Accessed on 19 August 2009.
- [100] wikipatterns.com. About Wiki Patterns. <http://www.wikipatterns.com/display/wikipatterns/About>.
- [101] xhtml.com. X/HTML 5 Versus XHTML 2. <http://xhtml.com/en/future/x-html-5-versus-xhtml-2/>, 2008.