

A dissertation Report On

# **Maintenance of Web Application and Sustaining Web Data Consistency**

Submitted in partial fulfilment of the requirements

For the award of the degree of

## **MASTER OF TECHNOLOGY IN SOFTWARE ENGINEERING**

By

Nishu Singh

(Roll No. 2K12/SWE/18)

Under the guidance of

**Mrs. Abhilasha Sharma**

Department of Software Engineering

Delhi Technological University, Delhi



**Department of Software Engineering**

**Delhi Technological University, Delhi**

**2012-2014**



**DELHI TECHNOLOGICAL UNIVERSITY**  
**CERTIFICATE**

This is to certify that the project report entitled **Maintenance of Web Application and Sustaining Web Data Consistency** is a bona fide record of work carried out by Nishu Singh (2K12/SWE/18) under my guidance and supervision, during the academic session 2012-2014 in partial fulfilment of the requirement for the degree of Master of Technology in Software Engineering from Delhi Technological University, Delhi.

Mrs. Abhilasha Sharma

Assistant Professor

Department of Software Engineering

Delhi Technological University

Delhi-110042



## DELHI TECHNOLOGICAL UNIVERSITY

### ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Mrs. Abhilasha Sharma**, Assistant Professor, Department of Software Engineering, Delhi Technological University for her valuable guidance and support in all the phases from conceptualization to final completion of the project.

I wish to convey my sincere gratitude to **Prof. O. P Verma**, Head of Department, and all the faculties and PhD. Scholars of Computer Engineering Department, Delhi Technological University who have enlightened me during my project.

I humbly extend my grateful appreciation to my friends whose moral support made this project possible. Last but not the least; I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Nishu Singh

Roll No. 2K12/SWE/18

## **ABSTRACT**

The heterogeneous and dynamic nature of component making up a Web Application, the lack of effective programming mechanism and undisciplined development processes induced by the high pressure of a very shorty time-to-market, make Web Application maintenance a challenging problem. A relevant issue consist of reusing the methodology and exploring the opportunity of using Reverse Engineering and Reengineering to support effective Web Application Maintenance.

The report presents the approaches of Reengineering in Web that explain how Reengineering process can lead to evolution activity in legacy system. We suggested V-model for Reengineering process that presents the technologies and approaches for building new Web services from existing Web Application. Based on the different parameter, we elucidate comparative study in between software reengineering versus Web reengineering and Reverse engineering versus Reengineering.

Proposed STAR paradigm is used to define and implement a reengineering process that involves web applications and supporting tools. It offers a structured method for defining reengineering process and paradigm which provide systematic approach to reengineer any web application.

Introduced Dynamic Data Extraction Algorithm for the maintenance of Web Application, which extract the content of Web page structure and remove the noise from the dynamic Web page. Furthermore we apply Cost Constructive Model in Web Application which help us to calculate the development time and maintenance cost. Comparison of the maintenance cost with and without noisy dynamic data has been done in all three modes of Cost Constructive Model.

## List of Figure(s)

Figure 1. Evolution life cycle.....	3
Figure 2. Stages of Reengineering.....	5
Figure 3. Web re-engineering V Model.....	14
Figure 4. Descriptions of Stages for Web Reengineering V Model.....	18
Figure 5. Stages of Software Reengineering.....	21
Figure 6. Stages of Web Reengineering.....	22
Figure 7. Various Situations of Web Application.....	28
Figure 8. Third Generation Tools.....	29
Figure 9. Categories of Web Application.....	30
Figure 10. Steps of Reengineering Process.....	32
Figure 11. (a) And (b) The layout structure and vision-based content structure of an example page.....	38
Figure 12. Level based structure of an example page.....	39
Figure 13. Process flow of DDE algorithm.....	40
Figure 14. Page Layout.....	41
Figure 15. DOM Tree of the Page.....	41
Figure 16. Layout of Block C.....	42
Figure 17. DOM Tree of Block C.....	42
Figure 18. Table based Content Structure.....	42
Figure 19. Example of a Web page.....	45
Figure 20. Block extraction of sample example.....	45
Figure 21. Separator Detection Process.....	46
Figure 22. Process flow of noise removal.....	48
Figure 23. Performance comparison of the DDE Algorithm.....	49
Figure 24. Performance comparison through the graph.....	50
Figure 25. Output window of DDEA.....	51
Figure 26. Result after sorted the output file.....	51
Figure 27. Result of extracted Dynamic data.....	53
Figure 28. Input the file for noise removal.....	54
Figure 29. Display the result Noisy Data.....	54

Figure 30. Web page size and extracted noisy data size.....	55
Figure 31. Graphical result of Web page size and extracted noisy data size.....	55
Figure 32. Snapshot of the Mat lab Program.....	58
Figure 33. Three modes of COCOMO model.....	58
Figure 34. COCOMO Coefficient.....	59
Figure 35. Calculation of development cost, development time and maintenance cost.....	60
Figure 36. Graphical result of development time and LOC.....	60
Figure 37. Graphical result development cost and LOC.....	60
Figure 38. Graphical result of maintenance cost and development cost.....	61
Figure 39. Calculation of metrics for Noisy Web page.....	61
Figure 40. Graphical result of development cost and maintenance cost for Noisy Web page.....	62
Figure 41. Graphical result of LOC, development cost and maintenance cost.....	62
Figure 42. Calculation of metrics for semi-detached.....	63
Figure 43. Graphical result of LOC, development cost and maintenance cost.....	63
Figure 44. Calculation of metrics for noisy web page.....	64
Figure 45. Graphical result for noisy Web page.....	64
Figure 46. Calculation of metrics for Embedded Mode.....	65
Figure 47. Graphical results for LOC, development time and maintenance cost.....	65
Figure 48. Calculation of metrics for noisy Web page.....	66
Figure 49. Graphical result of development cost and maintenance cost.....	66
Figure 50. Comparison of maintenance cost for organic mode.....	67
Figure 51. Graphical result of comparison of maintenance cost.....	67
Figure 52. Comparison of maintenance cost for semi-detached mode.....	68
Figure 53. Graphical result of maintenance cost.....	68
Figure 54. Comparison of maintenance cost for embedded mode.....	69
Figure 55. Graphical result of maintenance cost.....	69
Figure 56. Extended DDE Algorithm.....	71
Figure 57. Changes Importance example.....	75

## List of Table(s)

Table 1. Comparison of Reverse Engineering v/s. Reengineering.....	19
Table 2. Comparison of Software Reengineering v/s Web Application reengineering.....	24
Table 3. Different types of Reengineering Processes.....	33
Table 4. Heuristic rules in Block Extraction phase.....	44
Table 5. Different rules for different DOM nodes.....	44

## Table of Content

<b>Certificate</b> .....	II
<b>Acknowledgement</b> .....	III
<b>Abstract</b> .....	IV
<b>List of Figure(s)</b> .....	V-VI
<b>List of Table(s)</b> .....	VII
<b>Chapter 1</b>	
1. Introduction.....	2
1.1 Maintenance of Web based Application.....	2
1.1.1 Reengineering.....	4
1.1.1.1 Nature and Scope.....	5
1.1.1.2 Reverse Engineering.....	5
1.1.1.3 Transformational and Transfiguration.....	7
1.1.1.4 Forward Engineering.....	7
1.2 DDE Algorithm for Web Application and Sustaining consistency.....	7
1.2.1 Need of Algorithm.....	7
1.2.2 Sustaining consistency using DDE Algorithm.....	8
1.3 Structure of Thesis.....	8
<b>Chapter 2</b>	
2. Related Work.....	10
<b>Chapter 3</b>	
3. Overview of Web Reengineering.....	13
3.1. V- model for Reengineering.....	18
3.2. Comparative study of Reverse engineering and Reengineering.....	22
<b>Chapter 4</b>	
4. STAR Paradigm for Reengineering of Web Application.....	27
4.1. Situation.....	27
4.2. Tools.....	28
4.3. Application.....	29
4.4. Re-Structuring / Re-Conceptualization.....	31
<b>Chapter 5</b>	



5. Maintainability of Web Application by DDE Algorithm.....	36
5.1. Content structure of Web page.....	38
5.2. Dynamic Data Extraction Algorithm.....	40
5.2.1 Block Extraction.....	42
5.2.2 Separator Detection.....	45
5.2.3 Content structure construction.....	47
5.3. Noise removal from Web page using DDE Algorithm.....	48
5.4. Performance evaluation of DDE Algorithm.....	49
5.5. Result of DDE Algorithm.....	50
5.6. Empirical result.....	54
5.7. Cost Construction Model in Web Application.....	56
5.7.1. Calculation of Web size metrics.....	56
5.7.2. Calculation of effort, development time and maintenance cost.....	58
5.7.2.1. Organic Mode.....	59
5.7.2.2. Semi-Detached Mode.....	63
5.7.2.3. Embedded Mode.....	65
5.7.3. Comparison of maintenance cost.....	67
5.8. Optimizing Web Consistency using Visual Page Analysis.....	70
5.8.1. Visual Page Segmentation.....	70
5.8.2. Change Detection.....	71
5.8.3. Change Importance.....	72
<b>Conclusion.....</b>	<b>74</b>
<b>References.....</b>	<b>76</b>

# INTRODUCTION

# Introduction

---

The technological evolution of the last few years, has made the Web Service of the ideal platform for the appropriate support for their delivery and the development of Web-based applications. According to researchers [1] and [2], the Web application development is a multi-faceted Activity, that involves not only technical but also many other issues like organizational issue, managerial issue, even social and artistic issues. Web application development, refers a set of activities which are applied in order to develop a web application of high quality characteristics, and to accomplish this development efficiently and coherently. The popularity of web application has risen exponentially in last decades and every day, number of user increasing rapidly so that maintenance of web application is more concern now days. Web engineering becoming an important topic and is gaining more attention. It is fast growing area, not existing from centuries. This chapter gives the brief description of Maintenance of Web based Application, achieve maintainability using Dynamic Data Extraction algorithm and problem statement.

## **1) Maintenance of Web based Application**

Web maintenance and web reengineering both falls in the scope of web engineering. The World Wide Web has ability to ubiquitously provide and collect knowledge to the economy globalization together with the need of new marketing strategies has enormously boosted the development of Web Applications (WA). Software application is the backbone of the WWW infrastructure.

Most of the web applications, are developed under the proper schedules and in a rapidly developing environment. The development is often ad-hoc in nature through which the applications are structured and documented in poor fashion. It is very problematic to Maintenance such applications and increases the complexity of growth of the web application. Researchers have identified the need to re-engineer the system already existing web application into abstract design models. Creating appropriate design and architecture models is the solution to managing this complexity and supporting evolution of web applications.

Web application are multipurpose service provider. Web application must cope with an extremely short development evolution life cycle.

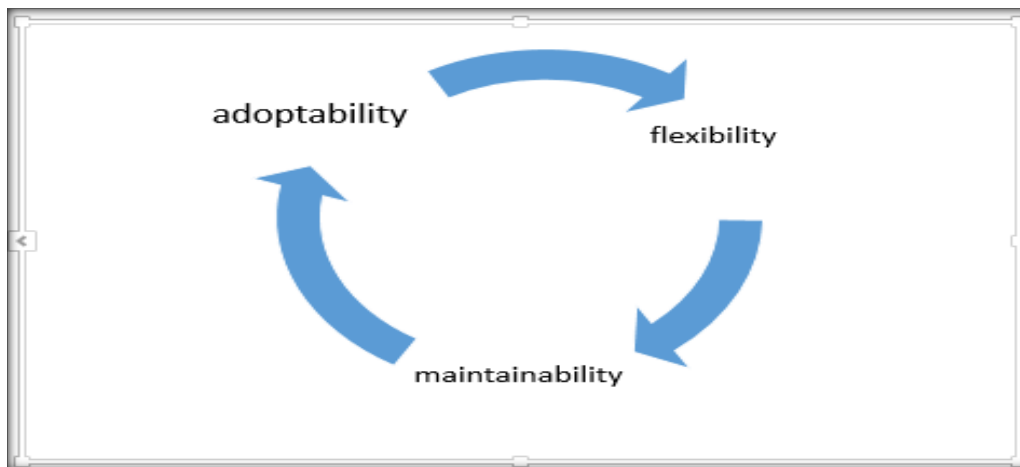


Figure 1. Evolution life cycle

A high level of flexibility, maintainability, and adaptability are actually necessary to compete and survive to market inflation. Unfortunately, to accomplish tight delivery schedules of web services, the web applications are usually directly implemented applications without producing any useful documentation for their evolution and maintenance, and so some of the requirements are never be satisfied by the customers. In order to satisfy a growing market request for Web applications and to deal with their increased technological complexity, we require specific methods and techniques able to support a disciplined and more effective development process. However, the high time pressure often forces the developers to implement the code of the application directly, without using disciplined development process, and this may have black effects on the delivered quality and documentation of the Web application. This situation is same as one occurring for traditional software produced in a short time, without respecting software engineering principles and using no disciplined development process. The quality and documentation is in poor fashion and it due to the following factors:

- Abortive and expensive maintenance,
- Unattainability of applying more structured and
- Documentation-based approaches.

Reverse engineering methods, techniques and tools have proved useful to support the post-delivery life-cycle activities of traditional software systems, such as maintenance and evolution.

Development of Web application refers a set of activities which applied in order to develop a web application of high quality having awaited characteristics, and to accomplish the development efficiently and coherently. It is a significant, fast developing area, which gaining more attention these days. Web maintenance and web reengineering both falls in the scope of web engineering. Web engineering is the discipline, which covers the overall cycle for the maintenance and development of web application. The sub section explains the Reengineering, need of Reengineering, nature and scope of Reengineering.

### **1.1.1 Reengineering**

Reengineering is the analysis of existing software system and modifying the existing software system it to constitute into a new form of system. Chikofsky and Cross define reengineering as ‘the examination and alteration of a subject system to reconstitute it in a new form and subsequent implementation of that form’ [3]. According to IEEE Std. 1998 ‘A system changing activity that results in creating a new system that either retains or does not retain the individuality of the initial system’ [4].

#### *Why Reengineering?*

Most of applications are developed under proper schedules and in a rapidly evolving environment. The development is often ad-hoc in nature and the applications are poorly structured and documented. Maintenance of such applications becomes problematic and increases the complexity in the application growth. Creating appropriate design and architecture models is the solution by managing this complexity and supporting evolution of applications. Researchers had identified the need to reengineer the system for already existing web applications into abstract design models.

When maintenance cost is not feasible, we go for reengineering the system. Reengineering makes the system new. It includes both the concept of forward engineering and reverse engineering.

#### *Need of Reengineering*

When system changes are confined to one subsystem, the subsystem needs to be reengineered, software and hardware support becomes obsolete and tools to support restructuring are readily available.

## Nature and Scope of Reengineering

When maintenance cost is not feasible, we go for reengineering the software system. Reengineering makes the software system new. Reengineering has the following three stages.

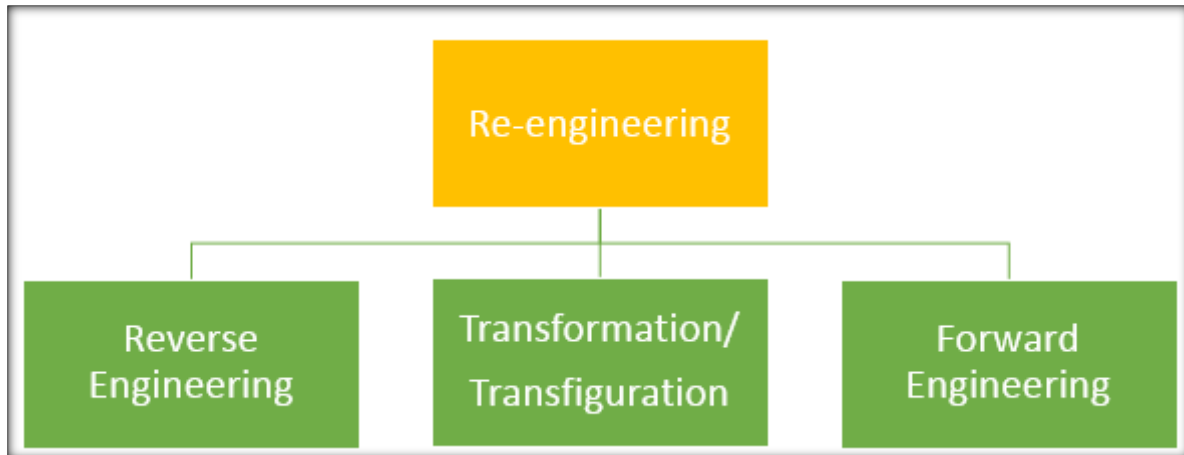


Figure 2. Stages of Reengineering

### 1.1.1.2 Reverse Engineering

Reverse engineering is the process of analysing the system which helps in recovering its design and specification. Reverse engineering is different from re-engineering. Reverse engineering is a process of analysis to determine the relationship of the system component and create the components of the system in another form or in a higher level of abstraction. The program itself is unchanged by the reverse engineering process. The objective of reverse engineering is to obtain the design or specification of a system from its source code although the objective of re-engineering is to produce a more maintainable system. Reverse engineering is used to produce a better system and it is a part of the re-engineering process. Reverse engineering is used during the re-engineering process to recover the program design specification which engineers use to help them understand a program before re-constructing its structure. However propose of a reverse engineering process for web encompassing the following phases:

1. Static Analysis
2. Dynamic Analysis

### Static Analysis

The number and the importance of Web applications have increasing rapidly over year by year. At the same time, the quantity and impact of security vulnerabilities in such applications have

grown as well [5]. Web application security is accomplished by static analysis and runtime analysis. Web application security has been a great challenge for this static analysis tool such as ASPWC is used to detect the attack and vulnerabilities based on taint analysis [6]. Static analysis does not require the execution of the application. It recovers web application architecture components and the static relations among them. HTML files, directory structure, scripting language sources as well as any other static information (e.g., database connections, use of applets/servlets) are processed. HTML pages and page sub elements (frames, forms, widgets) composing the given page are localized, classified and recorded in an intermediate representation. Central to the reverse engineering process is the mapping between web application elements and object oriented entities, according to Conallen proposals [7] [8] [9].

### **Dynamic analysis**

The dynamic analysis phase relies on the static analysis results. The web application is executed and dynamic interactions among the components are recorded. It is performed observing the execution of the web application, *tracing* to source code any event. Traced events are those observed by the user or related to components external to the web application (e.g., third party databases or WEB sites). Events are the HTML pages/frames/forms visualization, the submission of forms, the processing of data, a link traversal, or a database query, etc. All elements responsible of these actions (typically links, scripts, applets) are localized. The sequences of actions fired by an event, deriving from web application code control flow (e.g., access to a database following a user form submission) or from user actions (e.g., clicking on a link or submitting a form) are associated to sequences of messages exchanged between the objects of the web application.

In spanning of life cycle stages, reverse engineering covers a broad range starting from the existing implementation recapturing or recreating the design and deciphering the requirement actually implemented by the subject system. It includes re-documentation, design recovering from code, restructuring like code to code transformation. Reverse engineering is a process to transform a code into model through a mapping from a specific implementation language [10].

### **Purpose**

- Identify the system component and their interrelationships.
- Create representation of the system in another form or at a higher level of abstraction.
- Involves extracting design artifacts.

#### **1.1.1.3 Transformations and Transfiguration**

This phase involves the transition, alteration, modification, reformation, reconstruction, remodelling of the web application system. Web architecture is altered. It is modified, improved to cope with the new technology and new environment. It is the architecture designing stage. In web these transformation can be accomplished by changing in transaction design, adaptations of the code itself to the new computing platform, redesign of the UI to better suit the new constraints of the target platform.

#### **1.1.1.4 Forward engineering**

In this stage, we move from higher level of abstraction to low level. In this stage web application integrated according to new design. It is the traditional process of moving from higher level abstraction and logical implementation to independent design to the physical implementation of a system. It follows a sequence of going from requirement through designing its implementation.

### **1.2 DDE Algorithm for Web Application and Sustaining Consistency**

This section achieve the maintainability of web application by the adaptation of cost construction model in web application and explains how consistency of web pages is maintain using DDE. We introduce an algorithm to extract dynamic data of the web pages in the form of DOM (Dynamic Object Model), then store the dynamic data of web pages in database in the form of table. Consistency is maintained by the relational database, which consist insertion, updation and deletion on dynamic data.

#### **1.2.1 Need of Algorithm**

For last several years there is a growing trend in sharing a vast amount of information on the web. However with all this information new problem arises. Parsing and indexing it becomes a serious challenge. One of the main issues on the web is that the information contained on web sites is often mixed with irrelevant content, which might mislead automatic crawlers on the semantics of the page. A family of methods focused on how to deal with this issue has been developed and this paper brings summary of different approaches these methods are based on. The goal of these algorithms is to find segments of the web page, which contain the relevant information. They receive data and Meta data of a single web page as input and they output a content structure, usually in a form of semantic tree. The advantage of this family of methods



is that they are designed to work on a single page (unlike Template detection methods). That also implies the main disadvantage – scaling. Especially for visual methods processing a large number of web pages can become a serious problem.

### **1.2.2 Sustaining Consistency through DDE Algorithm**

In above section we explain the need of Dynamic Data Extraction Algorithm, which helps to extract the Web content structure (dynamic data of Web page), then remove the noisy data which include extra tags, unwanted data, duplicate data etc. and also display the noisy data. This algorithm extract the dynamic data in the form of table, if we store those extracted tables in the database then we can perform some operation from the database such as insertion, updation and deletion, also maintain the consistency. We will discuss this in detail later.

## **1.3 Structure of Thesis**

The remainder of the thesis is structured as follows:

*Chapter 2* gives the related work that has been done in the Web Application, Maintenance of Web Application, Dynamic Data Extraction Algorithm and Maintaining Consistency through the DDE Algorithm.

*Chapter 3* describes Web Reengineering which focus on unified V model of Web Reengineering with detailed Overview of Reengineering, Comparative study of Reverse & Reengineering and Software Reengineering & Web Reengineering.

*Chapter 4* presents STAR paradigms of Reengineering for Web Application and discuss the types of Reengineering.

*Chapter 5* presents the Dynamic Data Extraction Algorithm extension of VIPs algorithm with insight of Empirical results regarding Web page size and noise, Performance evaluation of DDE Algorithm & results of DDE Algorithm, Maintenance cost estimation using COCOMO model in Web Application and Maintaining Consistency.

The final section concludes the thesis and a discussion with possible future directions.

# RELATED WORK

The problem of analysing existing web application and web sites with the aims of maintaining, perceiving, testing them or assigning their quality has been addressed in some papers. New analysis approaches and tools, as well as adaptations of existing ones to the field of Web applications, have been proposed.

For example, **Hassan and Holt [11]** describe the modifications made to the Portable Bookshelf Environment (PSB), originally designed to support architectural recovery of large traditional applications, to make it suitable for the architectural recovery of a Web application.

**Martin et al. [12]** propose reusing the software engineering tool **Rigi [13]** as a means of analysing and visualizing the structure of Web applications. Other techniques and tools have been defined ad hoc for managing existing Web applications.

**Chung and Lee [14]** propose an approach for reverse engineering Web sites and adopt Conallen's UML extensions to describe their architecture. According to their approach, each page of the Web site is associated with a component in the component diagram, while the Web site directory structure is directly mapped into package diagrams.

**Ricca and Tonella [15] [16]** present the ReWeb tool to perform several traditional source code analyses of Web sites: they use a graphical representation of Web sites and introduce the idea of pattern recovery over this representation. The dominance and reachability relationships are used to analyse the graphs, in order to support the maintenance and evolution of the Web sites.

**Schwabe et al. [17]** define a framework for reusing the design of a Web application, by separating application behaviour concerns from navigational modelling and interface design.

**Boldyreff et al. [18]** propose a system that exploits traditional reverse engineering techniques to extract duplicated content and style from Web sites, in order to restructure them and improve their maintainability.

**Vanderdonckt et al. [19]** describe the VAQUISTA system that allows the presentation model of a Web page to be reverse engineered in order to migrate it to another environment.

**Di Lucca et al. [20]** present a method and a tool [21] for reverse engineering Web applications. The method defines the reverse engineering activities that are needed to obtain a set of views of a Web application at different abstraction levels. The views are cast into UML diagrams that

can be obtained with the support of a reverse engineering tool. Other approaches address Web application analysis with the aim of assessing or improving the quality of these applications. An analysis approach that allows the test model of a Web application to be retrieved from its code and the functional testing activity to be carried out is proposed in [22].

**Kirchner [23]** tackles the topic of accessibility of Web sites to people with disabilities, and presents a review of some tools available for checking Web pages for accessibility.

**Tonella et al. [24]** propose techniques and algorithms supporting the restructuring of multilingual Web sites that can be used to produce maintainable and consistent multilingual Web sites.

**Paganelli et al. [25]** describe the TERESA tool, which produces a task-oriented model of a Web application by source code static analysis, where each task represents single page functions triggered by user requests. The resulting model is suitable for assessing Web site usability and tracing the Web site user's profile.

**Shefali singhal et al. [26]** Web page segmentation is a technique which resolves this problem by logically dividing a web page into segments. These segments can be created by using DOM (Document Object Model) and VIPS (Visual Page Segmentation) techniques. In this paper, a hybrid method of web page segmentation has been designed using combination of DOM method and VIPS algorithm for developing segments from a web page. Here both the structural and visual aspects of a web page to create a segment have been considered. A segment is such a basic unit of web page which cannot be further divided.

**Anjali singh [27].** The goal of this paper is to explore the use of formal methods for filtration of noise from web pages. Filtration of noise from web pages is a difficult task which in turn leads to difficulty in segmentation. . Various automatic techniques use various algorithms of segmentation, which are mainly based on web source code (HTML) including template based analysis. Our insight is to use the DOM structures of web documents to efficiently implement a technique to remove irrelevant data, to optimize the WEB mining process .In this approach, we firstly build the Semantic Tree to partition the web page into the content parts/elements based on the web page tags.

**C.V.S.R SYAVASYA [28].** Several parameters come into use like Annual change traffic and development costs in man-months. Software is being used worldwide, in the same way maintenance of software is equally important as of development of software. Here, we have

taken particular range values of lines of code to evaluate costs pertaining to development which intern evaluates final maintenance costs.

## **CHAPTER 3**

---

# OVERVIEW OF WEB REENGINEERING

There are many definitions to describe the reengineering process, each are slightly different but all have same overarching theme, “the radical redesign of business process to achieve dramatic improvement in productivity and performance”. The two terms are radical which means getting rid of existing processes, procedure and inventing new ways and dramatic improvement means a quantum leap in performance [31]. Reengineering is the main process in which organization becomes more modernize and efficient and to meet demands for quality service, flexibility and low cost, process must made simple, it transform an organization in way that directly affect performance.

Web Re-engineering service helps to re-conceptualize and re-design your existing website and application service. Its objective is to re-structure the web application and its pages are modified to control and access standard and policies [34]. There are also 3 levels of the web re-engineering [35]:

- a) Design Recovery
- b) Analysis and Evaluation
- c) Redesign

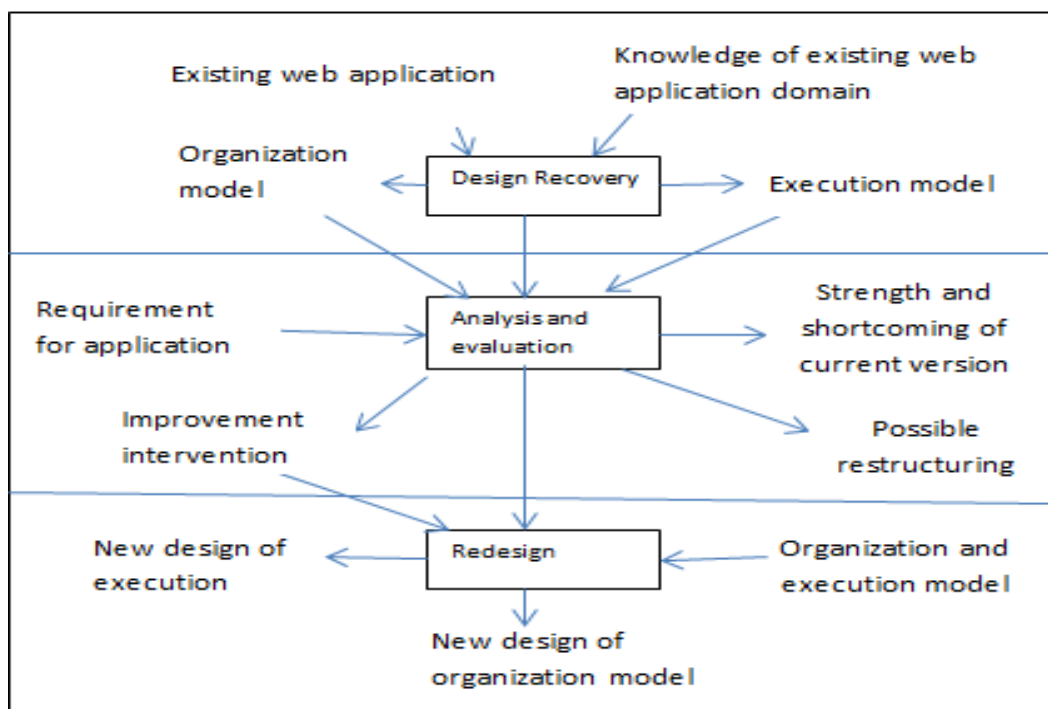


Figure 6. Stages of Web Reengineering

Design Recovery: For existing web site goal is to revise the models with data obtain by direct inspection and analysis of site's content and structure [36]. Model can be recreated using three steps of design recovery procedure:

1) Formalization the transaction (execution and organization model);

2) Creating the execution model;

3) Construction of organization model. Use the automated reverse engineering process to improve the efficiency of the process. Analysis and Evaluation: The result of design recovery procedure is to model a web application transaction using revised version of the execution and organization model. The next step of this process is to perform the user-oriented analysis and evaluation of the recovered execution and organization model. The objective of this step is to define the set of possible restructuring for the current design and implementation of the considered transaction addressing the strength and shortcoming highlighted by this level. Redesign: The objective of this level of re-engineering is to redesign the execution and organization model and introduces the changes defined during analysis phase into recovered the transaction design model produces new design.

As we already discussed that reverse engineering and forward engineering subset of the reengineering process, so that development community of model based approach has shown the interest for forward engineering with many models such as domain model, dialog model, presentation model and application model [37][38]. Today several environment support the forward engineering of web pages like Allegro Serve is a web server that dynamically generate and web enable existing application with a browser front end. However as some existing website were not built according to such approach so we adopt the reverse engineering process. The goal of reverse engineering is to recover the presentation and dialog model of the web pages [39]. In table 2 we have compared software reengineering with web reengineering with respect to the various perspectives like restructuring, retargeting, reverse engineering, forward engineering, data engineering, business process reengineering and architectural evolution.



<b>Parameters</b>	<b>Software Reengineering</b>	<b>Web application Reengineering</b>
<b>Restructuring</b>	Reorganize source code to perform some function more efficiently	Reorganize people, system and infrastructure to perform some basic functions in potentially more efficient ways.
<b>Retargeting</b>	Transport the source code and application system to new system	Adapt an existing business process to perform in new business functions
<b>Reverse Engineering</b>	Examine design of existing software system by deriving design from existing software code	Examine design of existing business process by extracting design from existing implementation
<b>Forward Engineering</b>	Develop new system design based on integration of new system requirements into existing system design.	Establish new business process design based on integration of new business requirement into existing business processes
<b>Data Reengineering</b>	Restructure the organization and format of stored information for use by software application.	Restructure the organization and format of stored information for use either more manual or automated processing activities.
<b>Architectural Evolution</b>	For software it generally requires centralised system is migrated to a distributed architecture it is essential that the core of that architecture should be a data management system that can be accessed from remote clients.	For web application it evolves through Client Server to N-TIER, Legacy to web Services, and Client Server to SOA (Service Oriented Architecture), legacy to web Enablement.

Table 2. Comparison of Software Reengineering v/s Web Application Reengineering

Web application must cope with an extremely short development evolution life cycle: A high level of flexibility, maintainability, and adaptability are actually necessary to compete and survive to market inflation. Unfortunately, to accomplish tight timing schedules to deliver web services, web applications are usually directly implemented without producing any useful documentation for their maintenance and evolution, and so those requirements are never be satisfied. In order to satisfy a growing market request for web applications and to deal with their increased technological complexity, we require specific methods and techniques able to

support a disciplined and more effective development process. However, the high time pressure often forces the developers to implement the code of the application directly, without using disciplined development process, and this may have black effects on the delivered quality and documentation of the web application. This situation same as one occurring for traditional software produced in a short time, without respecting software engineering principles and using no disciplined development process. Poor quality and poor documentation must be considered the main factors essentially abortive and expensive maintenance, unattainability of applying more structured and documentation-based approaches.

### **3.1 V-model for Web Reengineering**

The Re-engineered product goes through a complete web development life cycle and therefore it becomes mandatory for it to pass through complete testing cycle. The legacy system or product is transformed in new form by various means. The below figure illustrates the V-model for the Re-engineering process. A V-model as described below is proposed for designing the testing strategies for this category. Similar to the traditional V-model, left side of the Re-engineering V-model describes the stages of the design and coding and right side defines the corresponding stages of validation process. It has following phases:

#### **Phase 1: Requirement gathering for new web application**

The first step involves the collection of the new requirements. This will list out the key points why reengineering is required for the software under consideration. Client get start discussing with the web development team about the newly generated requirement due to market evolution, technology changes and for the product improvement for better performance. System is re-engineered in order to incorporate the new business requirements which involve functional and non-functional requirement. In this phase developer may make check list that deals with the various reason of re-engineering is required.

- Better performance
- Code restructuring
- New platform support
- Data migration
- New business requirement

- Requirement change

This phase require interviews with the client , mails and supporting documents given by the client, discussion notes, online chat, telephonic conversation, model sites/application etc.

Requirement analysis is carried out with new objective for re-engineering, cost involved in changes, supporting document and the approval.

Moreover the analysis should cover all the aspects on how the web application is going to join the existing system. The analysis should be done with in short time span having descriptive information. The analysis should be cost effective. To achieve this, the analyst should concern the designers, developers and testers to come up with an optimised work plan.

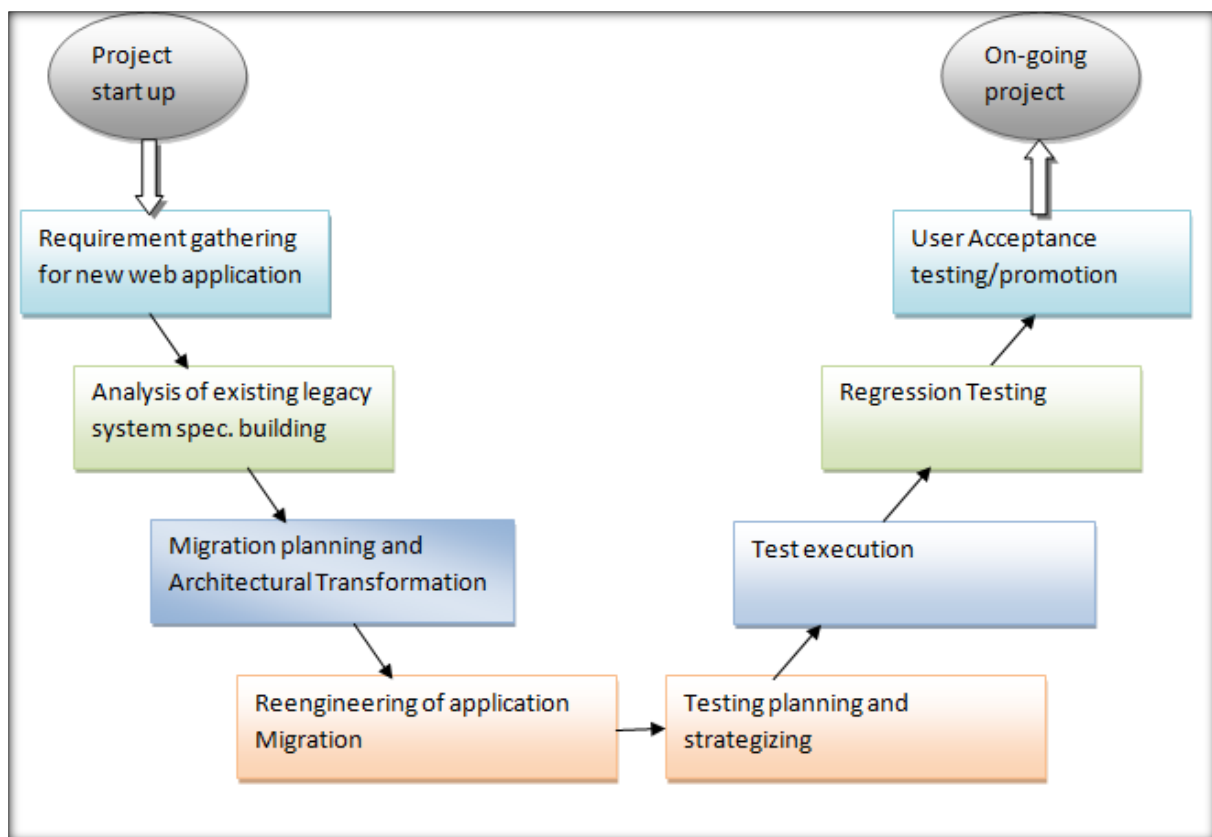


Figure 3. Web re-engineering V Model

## **Phase 2: Analysis of existing legacy system/specification building**

The second stage is the study of the legacy system functionality and underlying design and come out with the difference with new functions. The nature of re-engineering is to improve or transform existing system so it can be understood, controlled & reused as new system. Web re-engineering is vital to restore & reuse the things inherent in the existing system, put the cost of system maintenance to the lowest in the control & establish a basis for the development of system in future.

Web application broadly classify into two forms static application and dynamic application. static application implemented in HTML, & dynamic application provide client server interaction and consist of DHTML pages, JAVA server pages, java servlet, PHP, CGI, ODBC, JDBC etc. So in this phase we carefully analyse both perspective of web application (static & dynamic) and generate WAG graph for reusable components and objects for re-engineering. This phase may consist of three major steps

- Identification of reused Component
- Encapsulation of Identified Components to new system
- Analyse interfaces of the recovered components and define specification

Identification of legacy component aims to identify web components from legacy systems. The identified components should conform to specific user requirements that should relate to new functionality, access and manipulate data, are free of side effects, and comply with specific pre and post conditions. We encapsulate the recovered legacy system components to collections of object classes that encapsulate specific functionality of the legacy system. As an object encapsulates into legacy system flexible systems can be easily designed and implemented using the re-engineering paradigm. Furthermore, we analyse interfaces of the recovered components and define a specification to represent their interfaces. The service specification provides standard, enriched, and well-understood information about the interface and functionality of the offered services.

## **Phase 3: Migration planning and architectural transformation**

Migration planning that is how to move from the start to the Target Architectures by finalizing a detailed Implementation and Migration Plan. The objectives of migration planning Phase are to:

- Finalize the Architecture Roadmap and the supporting Implementation and Migration Plan.
- Ensure that the Implementation and Migration Plan is coordinated with the business approach to managing and implementing change in the business's overall change.
- Ensure that Transition Architectures is understood by key stakeholders
- Estimate Resource Requirements, Project Timings, cost estimation for introducing change.

#### **Phase 4: Re-engineering of application migration**

When existing systems become redundant, business switch from legacy systems to modern and new systems built on the latest technology / platforms. This switch is usually time consuming and expensive. A cost effective alternative to such scenarios is to reengineer, migrate or port the legacy systems into the latest technology / platforms.

*Application re-engineering:* In this we re-architect the product using new platforms and technologies such Web 2.0.

*Technology Migration:* This includes enterprises migrate applications to corporate standards and migration of products from older legacy technologies to newer technologies to ensure integration with other tools.

*Application Server Migration:* To take care of all cross-platform compatibility challenges, while the Client stays focused on product innovation.

*Database Migration:* This include migration of non-relational databases to industry-standard relational databases such as DB2, MS-SQL Server, Oracle, MySQL, thus increasing business agility.

*Migration of Middleware technologies:* It helps to migrate from legacy systems to new industry standard systems using implementation of middleware technologies (Server-side and Client-side) web services and others.

*Code restructuring:* To improve the coding paradigms it provides easy way of working. Code restructuring paid more attention in adding new features and re-factoring. It deals with continuously refactoring of web application which gives more flexible and maintainable code—Joshua Kerievsky, Refactoring to Patterns [2].

### **Phase 5: Test planning & strategizing**

The stage will include the test planning & test cases preparation if required as per the new requirements and strategizing the test execution for functional and non-functional areas. Test strategizing play an important role in carrying out the entire test execution program and involvement of high business risk, huge investments and mission critical systems make it important to strategizing the test phase. The best way is to identify the risky areas and the failure rate and then develop a test strategy

### **Phase 6: Test execution**

This stage carry out the functional test execution as per plan defined in the previous stage. Test execution carry out performance testing, if the major design changes are there or the new requirements are related to improvement of performance. It test all the links in web pages, database connection, forms used in the web pages for submitting or getting information from user, Cookie testing, Test for navigation, Content checking, Interface Testing include Web server and application server interface, Application server and Database server interface. Web testing includes functional testing, usability testing, interface testing, compatibility testing, performance and security testing. Performance testing is a used to determine the responsiveness, throughput, reliability, and scalability of a system under a given workload. Web application should sustain to heavy load. Web performance testing should include Web Load Testing and Web Stress Testing.

### **Phase 7: Regression testing**

Regression means retesting the effect of change in other parts of the web application. Test cases are executed again and again in order to check whether previous application functionality work appropriately and changes made have not introduced any new bugs or error. This testing can be performed on a new way to reconstructed system when new functionality added to it. A regression testing plan covers the updated functionalities. Many automated tool for regression testing is available for web application.

### **Phase 8: User Acceptance testing**

The purpose of this testing is to make sure that application meets the user's requirement and expectations. It ensures that the application is ready to deploy services and change has been

done effectively. The activities for user acceptance testing, ensure browser compatibility, checks that mandatory fields are given data in forms, check for field widths and time out , make sure that proper control is used to input data.

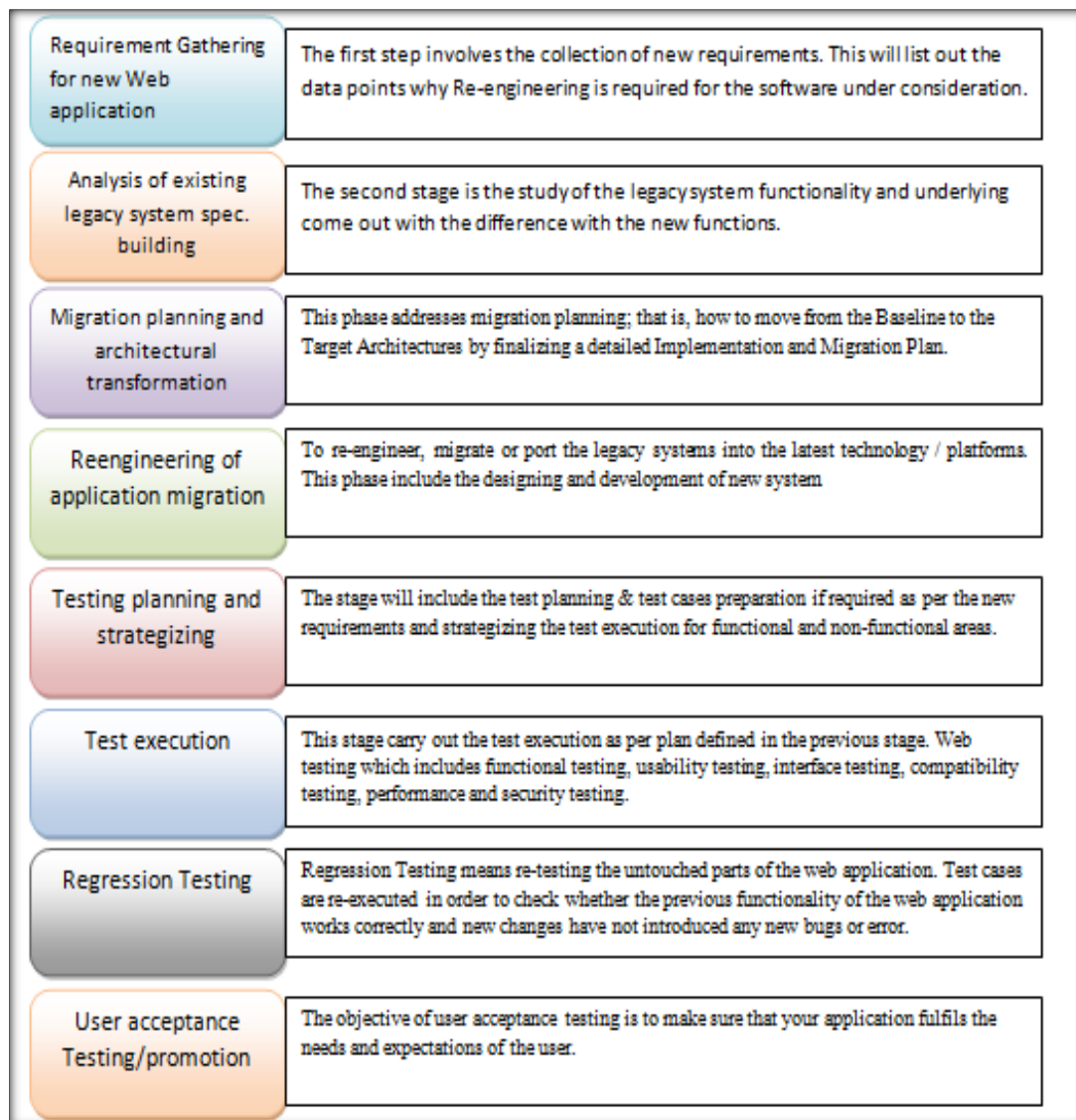


Figure 4. Descriptions of Stages for Web Reengineering V Model.

### 3.2 Comparative Study of Reverse engineering and Reengineering

We already explained the term Reverse engineering and Reengineering in previous section, where we found some differences between both the processes. This section focuses on some of the intrinsic and extrinsic differences of reengineering and reverse engineering [29]. Intrinsic comparisons: e.g. web page processing which were designed to be stateless whereas software

processing language make use of states. Extrinsic parameter: e.g. cost and results. Some of differences in the process of reengineering and reverse engineering are listed in table 1.

Parameters		Reverse engineering	Re-engineering
<b>Objective</b>		To drive the design or specification of a system from its source code.	To produce a new, more maintainable system.
<b>Definition</b>		Reverse engineering is finding out how a product works from the finished product.	Reengineering is examining the finished product and builds it again in better way.
<b>Process</b>	<b>Software Engineering</b>	<ol style="list-style-type: none"> <li>1. It is trying to recreate the source code from the compiled code and trying to figure out how a piece of software works given only the final system.</li> <li>2. It is important in software maintenance due to effectiveness &amp; analysing the consistency between design and implementation.</li> </ol>	<ol style="list-style-type: none"> <li>1. It is creating a new piece of software with similar functionality as an existing one but improving the way it was build.</li> <li>2. It is concerned with the reimplementatation of the legacy system to make them more maintainable.</li> </ol>
	<b>Web Engineering</b>	Reverse engineering extract information from the web application and allow more abstract representation (model) to reconstruct.	Reengineering reconstruct the model view which is extracted by reverse engineering and generating semantic and syntactic descriptions.
<b>Companies perspective</b>		Companies follow reverse engineering to copy and understand parts of a competitors products, which is illegal, to find out how their own product work in the event that the original plans were lost, in order to effect repair them.	Companies follow Reengineering to adapt generic product for a specific environment.
<b>Cost</b>		Continuous refactoring will decrease the total cost.	Cost is higher compared with reverse engineering.
<b>Result</b>		Reverse engineering improve the structure of existing.	Reengineering create the whole new system with different structure and different behaviour.

Table 1. Comparison of Reverse Engineering v/s. Reengineering

Table 1 reflects that reverse engineering is used during the software reengineering process to recover the program design, which engineers use for better understanding of a program before reorganized its structure. However, reverse engineering need not always be followed by reengineering. The activities of reengineering process involves source code translation which converts program from an old programming to modern version and in data reengineering the data processed by the program is change to reflect program changes [30], reverse engineering is used to analyse the program and information extracted from the program which helps in



document its organization and functionality and to improve program structure that controls the structure of program and modified to make it easier for understanding, program modularization is a part of reverse engineering that used to group together related parts of program.

engineering is used to analyse the program and information extracted from the program which helps in document its organization and functionality and to improve program structure that controls the structure of program and modified to make it easier for understanding, program modularization is a part of reverse engineering that used to grouped together related parts of program.

# **STAR Paradigm for Reengineering of Web Application**

# STAR Paradigm for Reengineering of Web Application

---

Re-engineering process is usually run to abstract and withdraw data, document them from existing software, and to unified these documents and data with expert knowledge and previous experiences that cannot be instinctively reconstructed from software. According to the STAR paradigm, a re-engineering process is characterized by situation, tools, application and renovating. Situation defines a set of views of the applications to be reverse engineered. Tools include techniques and tools to support the information recovery process. Application describes which particular types of applications require to migrate from one technology to another. Lastly restructuring specifies the actual implementation of re-engineering process and also decides which type of re-engineering process (transaction re-engineering, data re-engineering, graphic design re-engineering etc.) is suitable for above paradigm .Possible situation, tools, application and renovating characterizing Web application re-engineering processes are presented below.

## 4.1 Situation

In the field of Web applications, situation explains the possible scenario in which requirement of re-engineering has emerged. A reverse engineering process may aid assessment of the characteristics of an existing application, in order to be able to evaluate its quality attribute, including reliability, security or maintainability [40].

Several situations and scenarios are there to re-engineer a web application such as alteration in web pages by Percolating the objects, tags and elements of the web pages that include selection of any HTML item with given properties that require keeping all control mechanism and discarding the unwanted tags and elements from web pages.

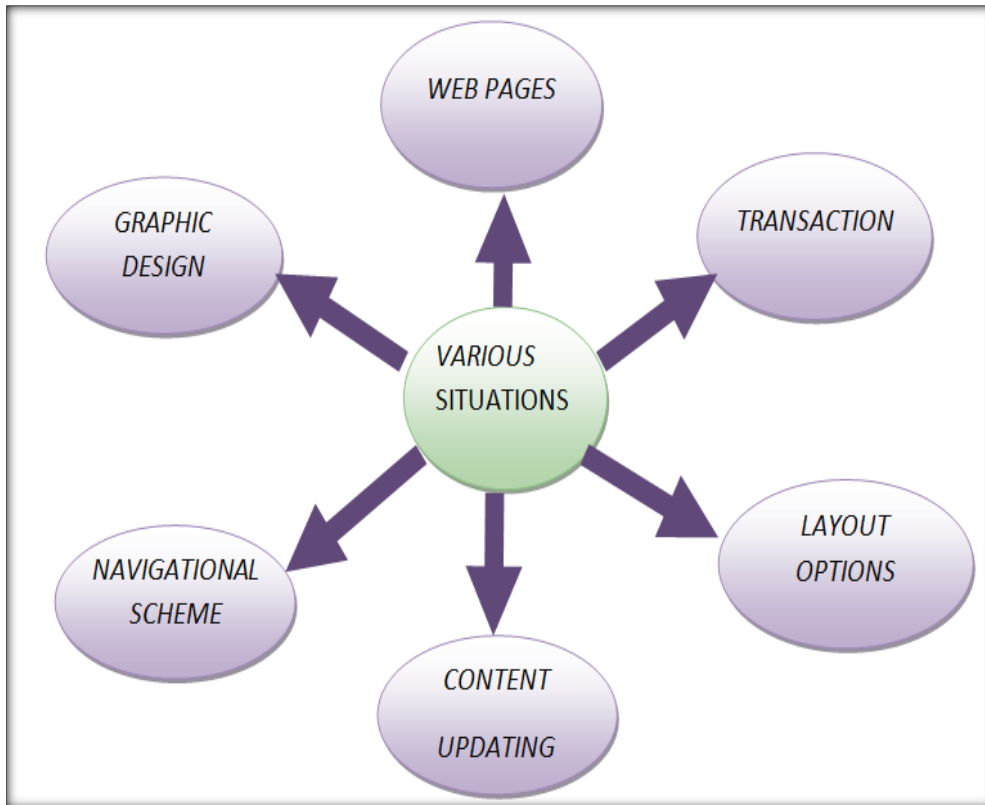


Figure 7. Various Situations of Web Application

Reengineering of web pages can be accomplished by detecting and analysing the interaction of objects and then transforming these objects for the adaption of new platform itself and generating the source code into new language. Transformation in the layout options and relationships that include alignment balancing which depend upon the position of the objects on the page and content updating of the Web Pages according to the requirement changes, market evolution, usage and owner of website. Re-engineering is used to adapt a UI to different format and to alter a navigational scheme that tells how web elements such as server page, client page, form, frame, email etc. are linked.

## 4.2 Tools

The recovery of information from an existing Web application and the production of models, documentation of its relevant features that cannot be effectively accomplished without the support of suitable techniques and Tools that automate the Web application analysis. However, the diverse and dynamic nature objects producing the application, and the lack of effective mechanisms for implementing the basic software engineering principles in Web applications, that makes analysis process more complex and make it necessary to address specific

methodological and technological problems. Some transcoding tools [41] [42] [24] [19] automatically transform a UI code from the original platform to a target platform.

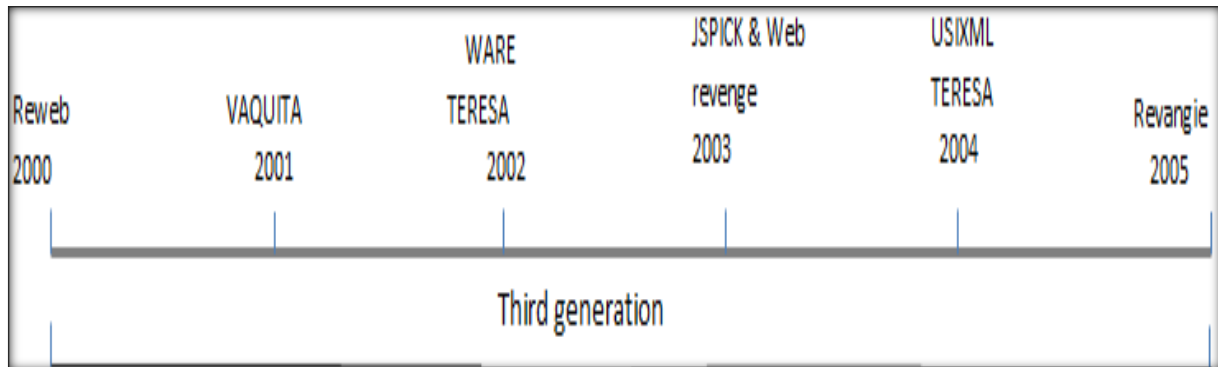


Figure 8. Third Generation Tools

More precisely, heterogeneous software components developed with different technologies and languages require techniques and tools for multi-language analysis. The existence of ‘dynamic software components’ in a Web application, such as pages created at runtime depending on user input, will impose the application of dynamic analysis techniques and static analysis of the code in order to obtain more precise information about the Web application behaviour. In addition, the absence of effective mechanisms for implementing the software engineering principles of modularity, encapsulation, and separation of concerns, will make the use of suitable analysis approaches, such as program slicing which is necessary in order to localize more cohesive parts in the Web application code. Finally, on the basis of the situations, tools / technique and application identified the sequence of activities composing the re-engineering process, their input/output and responsibilities will be precisely set out. The reverse engineering process can be executed with the support of various engineering tools proposed as reweb [19], vaquita [25], WARE [43] [44], tersa JSPICK [45], web revenge and revangie tool.

### 4.3 Applications

Web development within an organisation depends upon several factors. The number and the importance of Web applications have increasing rapidly over year by year. At the same time, the quantity and impact of security vulnerabilities in such applications have grown as well [46]. The motivation depends upon the initial purpose of Web usage, the customer’s expectations and the competitive environment. The drive to systematise development is subject to overall

view of the Web and conscious policy decisions within the organisation. For example, a low level view of the Web is lead to ad hoc. Initially we need to understand the problem domains that currently addressed by web. Table 3 presents categories of Web applications. Organizations that started their Web development early may also have followed a similar order in the past. Although, it is possible to start Web development with applications in any category, this table has been useful to explain to organisations with modest presence on the Web how they might improve or benefit from incremental exposure, thus keeping the risks to the minimum.

<b>Workflow</b>	Planning and scheduling systems, inventory management, status monitoring
<b>Collaborative work Environments</b>	Distributed authoring systems, collaborative design tools
<b>Informational</b>	Online newspapers, product catalogues, newsletters, service manuals, classifieds, e-books
<b>Interactive</b> <ul style="list-style-type: none"> <li>▪ User-provided information</li> <li>▪ Customized access</li> </ul>	Registration forms, customized information presentation, games
<b>Online communities, Marketplaces</b>	Chat groups, recommender systems, marketplaces, auctions
<b>Web Portals</b>	Electronic shopping malls, intermediaries
<b>Transaction</b>	E-shopping, ordering goods and services, banking
<b>Web Services</b>	Enterprise applications, information and business Intermediaries

Figure 9. Categories of Web Application

Migration of applications to the newer technologies can give business a leading edge by removing inefficient workflow and processes while preserving original objectives, model and investment. We can help enterprises in migration of the legacy systems from old technologies to present day platforms. Reengineering strategically designed to overcome the cross platform compatibility challenges. Due to upcoming advance technology and growing business states,

there is need for the migration of legacy software systems to new technologies and environments. There are different kind of legacy system re-engineering services that includes language and database migration, platform-to-platform porting and system redevelopment.

A web application must follows the enterprises standard and rules implemented in a legacy application, while transforming those to new business and architecture requirements, to produce a flexible, tested or validated modified system. Re-engineering and Migration Benefits are the saving time and effort, Enhancements in operational efficiency, Benefits of the latest technologies and platforms.

#### **4.4 Restructuring/Re-Conceptualization**

Reengineering is the analysis of existing software system and modifying it to constitute into a new form. Chikofsky and Cross define reengineering as ‘the examination and alteration of a subject system to reconstitute it in a new form and subsequent implementation of that form’ [47]. According to IEEE Std. 1998 ‘A system changing activity that results in creating a new system that either retains or does not retain the individuality of the initial system’ [48]. Techniques of static and dynamic analysis of the source code and dynamic data were taken into account.

Additional techniques for analysing the Web application structure and identifying relevant subsets of its components were also considered where clustering techniques were defined to carry out this analysis. Finally, the specifications of the tools required to support these analyses could be defined.

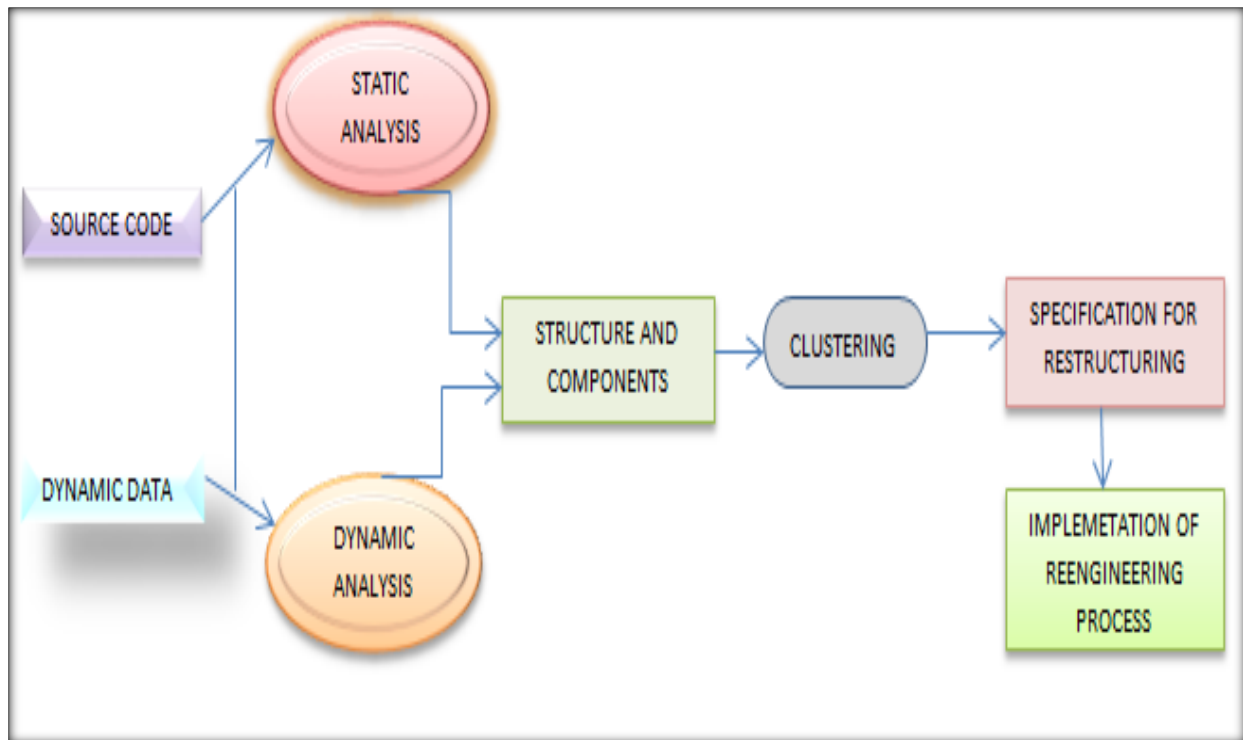


Figure 10. Steps of Reengineering Process

In the static analysis, all the information necessary to obtain the inventory of the Web application entities (pages and inner pages) such as forms, scripts and other web objects are identified, the static relations between them is extracted from the code, and the statements producing link, submit, redirect, build, and other relationships are identified and localized in the code. The dynamic analysis phase is based on and uses static analysis results. The Web application is executed and dynamic interactions among the entities described in the class diagram are recorded. Dynamic analysis is performed by observing the execution of the Web application, and tracing any observed event or action to the corresponding source code instructions. Analysis of the execution is a task that can be carried out either automatically on the basis of the application code or manually by observing the page execution by a browser and recording the observed events. The information about the Web application obtained by static and dynamic analysis can be used to produce a graph whose nodes represent the set of Web application entities, and whose edges describe the different relationships among these entities. In [49], this kind of graph has been named as WAG; Web Application connection Graph. WAG analysis may support the comprehension of the application. However, since this graph may be large (in terms of the number of nodes and edges) even in the case of small size Web applications, in order to simplify the analysis of large WAG graphs, some kind of



automatic clustering [49] can be used to decompose this graph into smaller cohesive parts. This clustering approach evaluates the degree of coupling between entities of the application (such as server pages, client pages and client modules) that are interconnected by Link, Submit, Redirect, Build, Load in Frame, and Include relationships. After getting more abstract specification from clustering algorithms actual implementation of re-engineering process is executed according to selected situation and application. Different type of re-engineering processes are generated based upon the above three perspective are listed below.

Data Reengineering	The process of analysis and re-organising the data structure and data values in a system to make it more understandable.
Transaction Reengineering	In transaction oriented website, the user executes a series of activities in order to carry out a specific task. Business processes are realized by means of transaction, in which this context can be interpreted as high level work flows corresponding to user tasks.
Application Migration	Migration application to the new technologies can give business leading edge by removing inefficient workflow and process while preserving original objectives, model and investment.
Graphic Design Reengineering	It is used to modify a user interface to different context. To change user interface developers do not necessarily want to start from scratch to design a UI for a new platform since UI already exists.
Reengineering of Web pages	Reengineering of web pages can be accomplished by detecting and analysing the interaction of object and transforming these objects for the adoption of new platform itself and generating the source code into new language.
Business Process Reengineering	It is the analysis and redesign of workflow within and between enterprises. It defines all the process in an organization and prioritizes them in order to redesign urgency.

Table 3. Different types of Reengineering Processes

Let us take an example illustrating real motive of STAR paradigm, suppose there is situation of graphic design need to be alter then we can have transcoding tool TERESA[50] that automatically transform a graphic interface code from the original platform to a target platform with any of the application suppose Interactive websites ( User-provided information Customized access) hence above three paradigm itself implies that restructuring can be performed using graphic design reengineering transfigure a final user interface into a logical representation that is allow forward engineering to port a UI from one computing platform to another with maximum flexibility and minimal effort.

**Maintainability of Web Application by Using  
DDE Algorithm**

# Maintainability of Web Application by Using DDE Algorithm

---

The chapter first focuses on the content structure of Web pages and DDE Algorithm, Where DDE stands for Dynamic Data Extraction. DDE Algorithm helps to reduce maintenance time of web application by extraction of dynamic data of Web pages and remove noisy data from web pages. Then we work on dynamic data and maintenance cost estimation using COCOMO (Constructive Cost) Model in Web Application.

Currently there is no Web data extraction method that performs satisfactorily on a wide domain of Web pages. Each of the current methods only works fine on the test data of their specific domain. If a third party data is used, usually the results of current methods are unacceptable. So an effective and general Web data extraction methods needs to be developed. The new algorithm developed which use semantic structure of Web page content together with structural analysis of the visual blocks to achieve better results.

All current Web page segmentation algorithms employ HTML tag information to partition a Web page into a set of blocks with each containing related information. The most popular ones are Document Object Model (DOM)-based segmentation [51], location-based segmentation [52] and visual-based segmentation [53].

In DOM-based segmentation algorithm, a Web page is first represented as a HTML DOM tree, then tags, such as p (for paragraph), table (for table), ul (for list) and h1-h6 (for heading) are used to building block. Because HTML DOM is provided for not only content organization but also layout presentation, it is often not accurate enough to use tags such as table and p to represent different semantic block in a Web page.

The location-based segmentation algorithm is based on the layout of a Web page. To be specific, a Web page is generally separated into 5 regions: top, down, left, right and centre. The problem of this algorithm are that such 5-regions layout template cannot be applied to all Web pages and segmentation is too rough to exhibit semantic coherence.

The VIPS algorithm, various visual cues, such as position, font, colour and size are taken into account to achieve a more accurate content structure on the semantic level. All current segmentation algorithms cannot determine the data regions or data record boundaries because

they are not developed for this purpose, but they provide the important semantic partition information of Web page.

In this chapter, we given a Dynamic Data Extraction algorithm (DDE) to extract the semantic structure for a web page. Such semantic structure is a hierarchical structure in which each node will correspond to a block. Each node will be assigned a value (Degree of Coherence) to indicate how coherent of the content in the block based on visual perception. The DDE algorithm makes full use of page layout feature: it first extracts all the suitable blocks from the html DOM tree, then it tries to find the separators between these extracted blocks. Here, separators denote the horizontal or vertical lines in a web page that visually cross with no blocks. Finally, based on these separators, the semantic structure for the web page is constructed. DDE algorithm employs a top-down approach, which is very effective. It first extract all suitable nodes from the HTML DOM tree internally, and then finds the separators between these nodes. The procedure that DDE segments a web page into visual blocks are as follows:

### **1. Block Extraction**

The Web page is recursively segmented into visual blocks based on visual cues. A Degree of Coherence (DoC) tells how close the contents within the visual block related to each other. DoC value is assigned to each visual block. When the DoC value is under a threshold, the visual block will be consider as a leaf block.

### **2. Separator Detection**

Horizontal and vertical separators are identified from the Web pages and weight is set for each separator.

### **3. Content Structure Construction**

Start from the lowest weight separator, the blocks around the separator begin to merge, the DoC of the new block is set as the maximum weight of the separators in it. The merge process iterates until the highest separator is met. Then each node is checked to see if its DoC is under threshold. If there is any leaf node having a DoC larger than the threshold, the node will be segmented to smaller blocks. Then step 2, 3 will be repeated until all end nodes have DoC value lower than the threshold.

After these 3 steps all the block which is extracted from the web pages are stored in the form of table. Table is made with the help of segment value, all these table are stored in a file. Explain these steps later in detail with example.

## 5.1 Content Structure of Web Page

Similar to [51], we define the *basic object* as the leaf node in the DOM tree that cannot be decomposed any more. In this paper, we propose the vision-based content structure, where every node, called a *block*, is a basic object or a set of basic objects. It is important to note that, the nodes in the vision-based content structure do not necessarily correspond to the nodes in the DOM tree. Similar to the description of document representation in [54], the basic model of *vision-based content structure* for web pages is described as follows.

A web page  $\Omega$  is represented as a triple  $\Omega = (O, \Phi, \delta)$ .  $O = \{\Omega^1, \Omega^2, \Omega^3 \dots \Omega^N\}$  is a finite set of blocks. All these blocks are not overlapped. Each block can be recursively viewed as a sub-webpage associated with sub-structure induced from the whole page structure.  $\Phi = \{\phi^1, \phi^2 \dots \phi^T\}$  is a finite set of separators, including horizontal separators and vertical separators. Every separator has a weight indicating its visibility, and all the separators in the same  $\Phi$  have the same weight.  $\delta$  is the relationship of every two blocks in  $O$  and can be expressed as:

$$\delta = O * O \rightarrow \Phi \cup \{NULL\}.$$

For example, we take a yahoo shopping website in below figure.

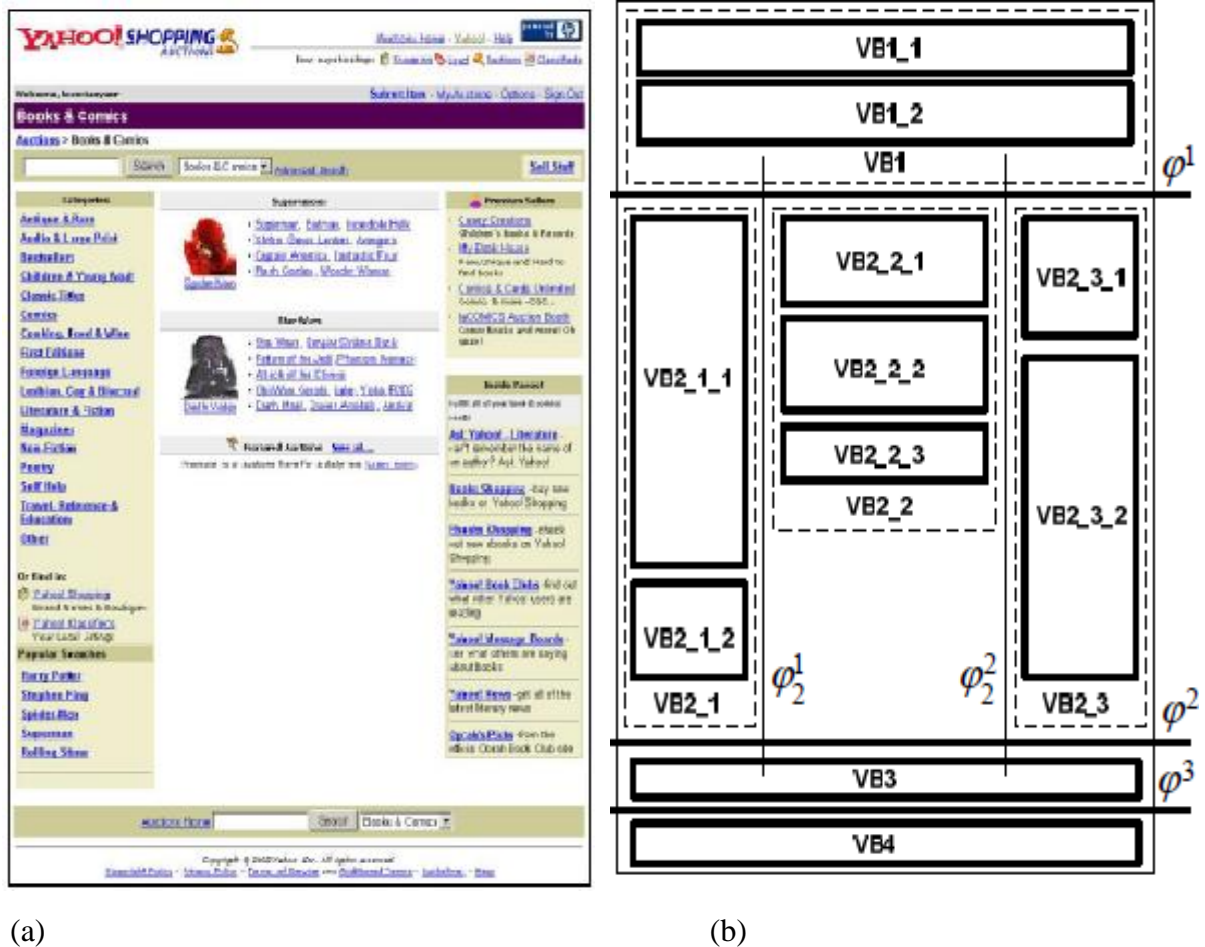


Figure 11 (a) and (b) The layout structure and vision-based content structure of an example page.

Figure 2 shows the segmented form of web page with assigned weight which helps to extract the segments of web page and converted into tree form.

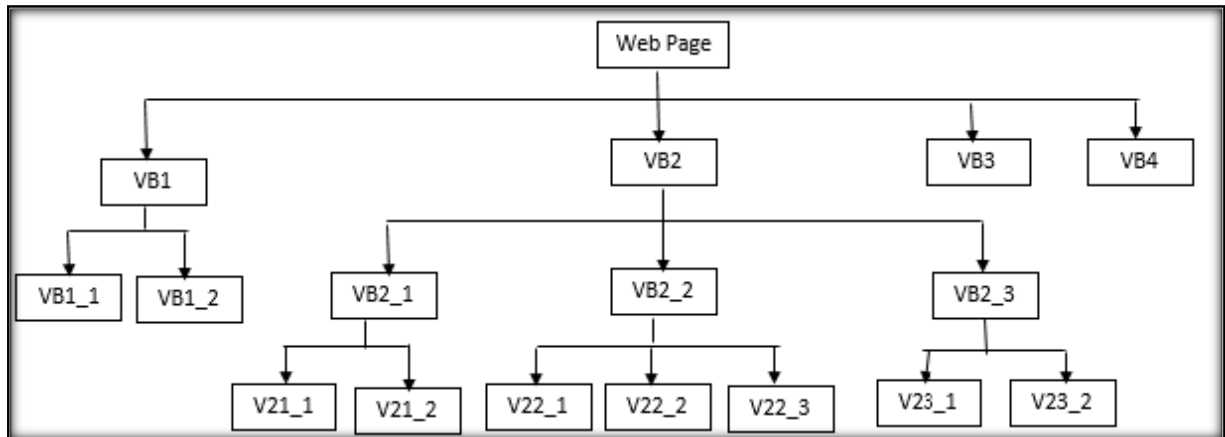


Figure 12 Level based structure of an example page

For each visual block, the *Degree of Coherence* (DoC) is defined to measure how coherent it is. DoC has the following properties:

- The greater the DoC value, the more consistent the content within the block;
- In the hierarchy tree, the DoC of the child is not smaller than that of its parent.

In our algorithm, DoC values are integers ranging from 1 to 10, although alternatively different ranges (e.g., real numbers, etc.) could be used. We can pre-define the *Permitted Degree of Coherence* (PDoC) to achieve different granularities of content structure for different applications. The smaller the PDoC is, the coarser the content structure would be. For example in Figure 1(a), the visual block VB2\_1 may not be further partitioned with an appropriate PDoC. Different application can use VIPS to segment web page to a different granularity with proper PDoC. The vision-based content structure is more likely to provide a semantic partitioning of the page. Every node of the structure is likely to convey certain semantics. For instance, in Figure 11(a) we can see that VB2\_1\_1 denotes the category links of Yahoo! Shopping auctions, and that VB2\_2\_1 and VB2\_2\_2 show details of the two different comics.

## 5.2 Dynamic Data Extraction Algorithm

Dynamic Data Extraction algorithm (DDE) to extract the semantic structure for a web page. The process flow of DDE algorithm is given below.

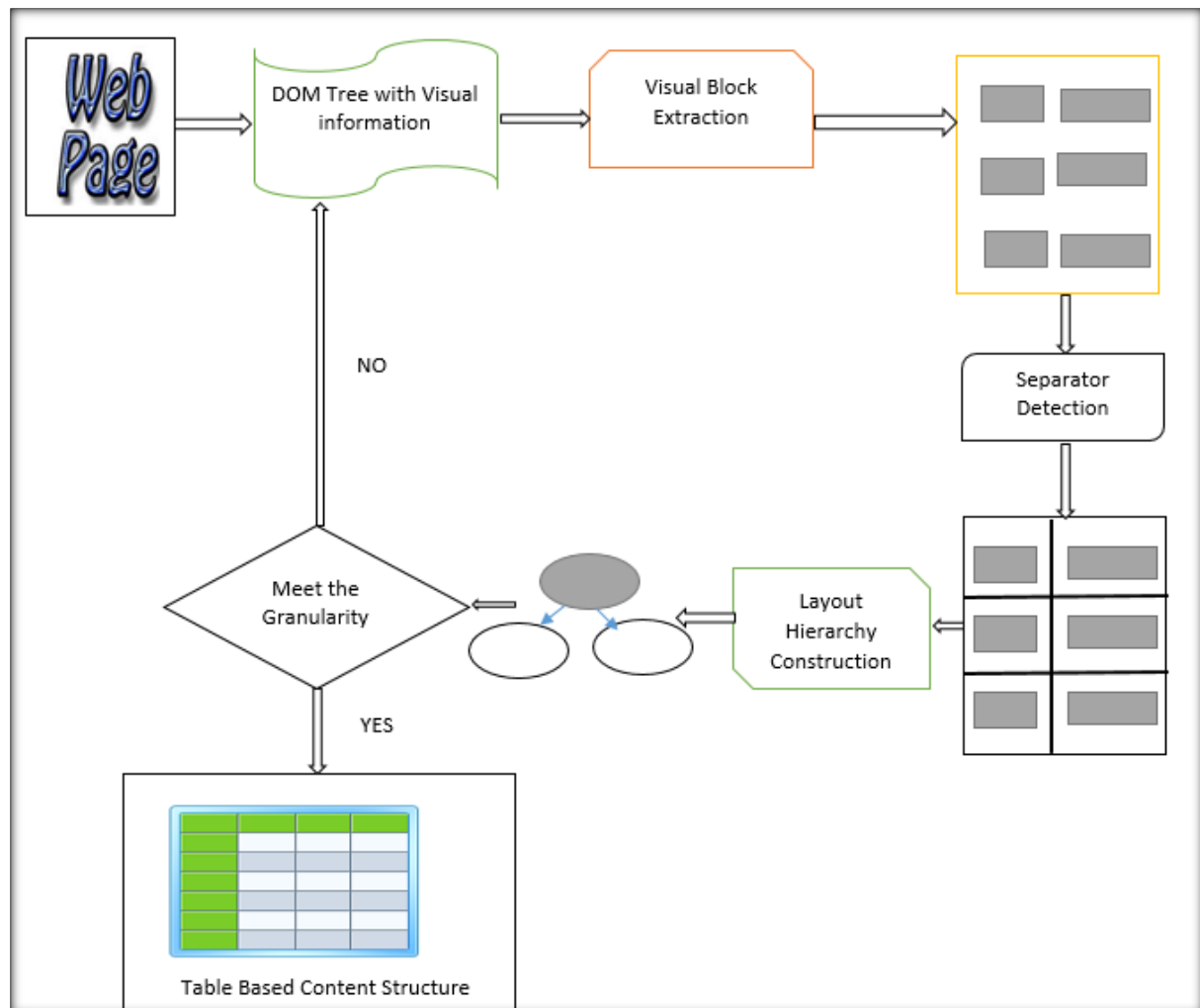


Figure 13 Process flow of DDE algorithm

In this section, we introduce our DDE algorithm. Basically, the content structure of a page is obtained by combining the DOM structure and the visual cues. The segmentation process has three steps: block extraction, separator detection and content structure construction. These three steps as a whole are regarded as a round. The algorithm is top-down. The web page is firstly segmented into several big blocks and the hierarchical structure of this level is recorded. For each big block, the same segmentation process is carried out recursively until we get sufficiently small blocks whose DoC values are greater than pre-defined PDoC.

For each round, the DOM tree with its visual information corresponded to the current block (page for the first round) is obtained from a web browser, as we show in figure 14. Then, from



the root node(s) of the DOM tree (e.g. DOM node 1 in Figure 15), the block extraction process is started to extract blocks from the DOM tree based on visual cues. Every DOM node (node 1, 2, 3, 4, 5, 6, 7 as shown in figure 15) is checked to judge whether it forms a single block or not. If not (node 1, 3, 4 in figure 15), its children will be processed in the same way. We will assign a DoC value to each extracted block (node 2, 5, 6, 7 in figure 15) based on the block's visual property.

When all blocks of the current round in current page or sub-page are extracted, they are put into a pool. Separators among these blocks are identified and the weight of a separator is set based on properties of its neighbouring blocks. The layout hierarchy was constructed based on these separators. After constructing the layout hierarchy of the current round, each leaf node of the content structure is checked to see whether or not it meets the granularity requirement. If not, this leaf node will be treated as a sub-page and will be further segmented similarly.

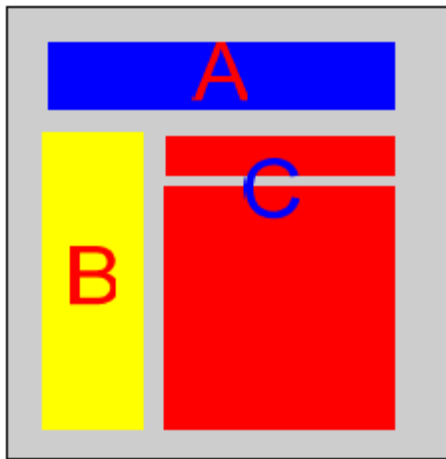


Figure 14 Page Layout

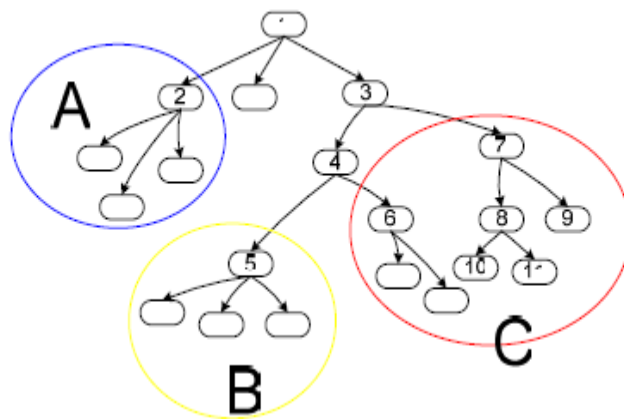


Figure 15 DOM Tree of the Page

For example, if the Block C in figure 15 does not meet the requirement, we treat this block as a sub-page and it will be further segmented into two parts, C1 and C2, as shown in figure 16 and figure 17.

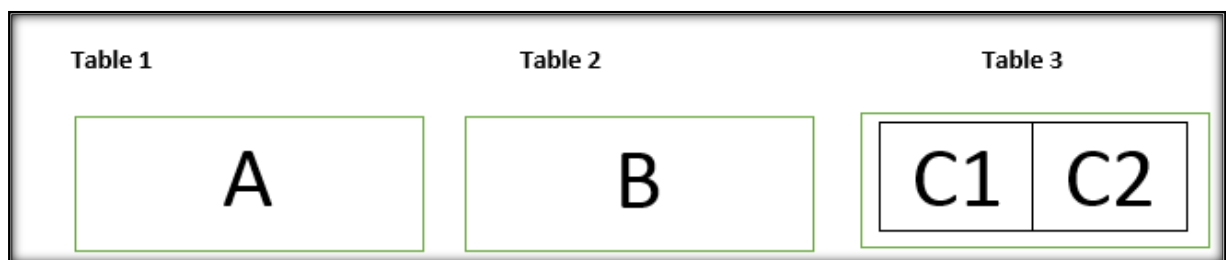
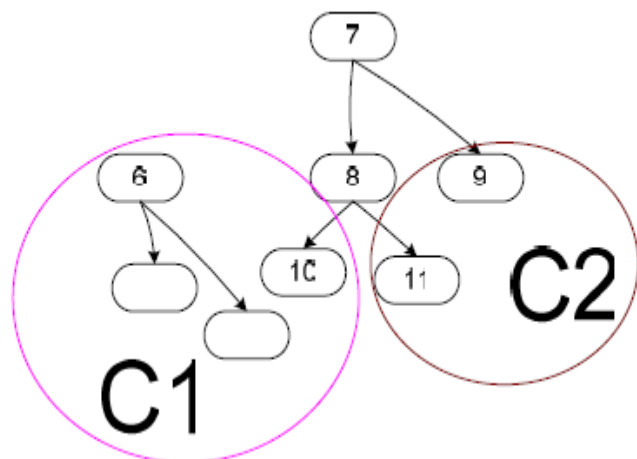
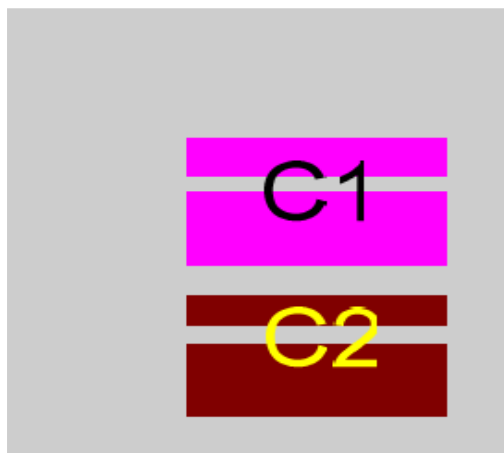


Figure 18 Table based Content Structure

After all blocks are processed, the final Table-based content structure for the web page is outputted. In the above example, we finally obtain a Table-based content structure tree as shown in Figure 5. In the following three subsections, we will describe the block extraction, separator detection and content structure construction, respectively.

### 5.2.1 Block Extraction

In this step, we aim at finding all appropriate visual blocks contained in the current sub-page. In general, every node in the DOM tree can represent a visual block. However, some “huge” nodes such as `<TABLE>` and `<P>` are used only for organization purpose and are not appropriate to represent a single visual block. In these cases, the current node should be further divided and replaced by its children. Moreover, due to the flexibility of HTML grammar, many web pages do not fully obey the W3C HTML specification, the DOM tree cannot always reflect the true relationship of the different DOM node. For each extracted node that represents a visual

block, its DoC value is set according to its intra visual difference. This process is iterated until all appropriate nodes are found to represent the visual blocks in the current sub-page.

Internally the DOM node can be divided based on following considerations:

- The properties of the DOM node itself. For example, the HTML tag of this node, the Background colour of this node, the size and shape of this block corresponding to this DOM node.
- The properties of the children of the DOM node. For example, the HTML tags of Children nodes, background colour of children and size of the children. The number of Different kinds of children is also a consideration

Some important cues which are used to produce heuristic rules in the algorithm are:

- Tag cue:
  1. Tags such as <HR> are often used to separate different topics from visual perspective. Therefore we prefer to divide a DOM node if it contains these tags.
  2. If an *inline node* has a child which is *line-break node*, we divide this inline node.
- Colour cue: We prefer to divide a DOM node if its background colour is different from one of its children's. At the same time, the child node with different background colour will not be divided in this round.
- Text cue: If most of the children of a DOM node are text nodes or virtual text node, we prefer not to divide it.
- Size cue: We predefine a relative size threshold (the node size compared with the size of the whole page or sub-page) for different tags (the threshold varies with the DOM nodes having different HTML tags). If the relative size of the node is smaller than the threshold then we prefer not to divide the node.

Based on these cues, we can produce heuristic rules to judge if a node should be divided.

If node should not be divided, a block is extracted and we set the DoC value for this block. We list the heuristic rules in below table.

<b>Rule 1</b>	If DOM node is not a text node and it has no valid children, then this node cannot be divided and will be sent
<b>Rule 2</b>	If DOM node has only one valid child and the child is not a text node, then divide this node
<b>Rule 3</b>	If DOM node is a root node of the sub-DOM tree and there is only one sub DOM tree corresponding to this block, divide this node.
<b>Rule 4</b>	If all the child nodes of the DOM nodes are text nodes or virtual text nodes, do not divide the node.
<b>Rule 5</b>	If one of the child nodes of the DOM node is line-break node, then divide this DOM node.
<b>Rule 6</b>	If one of the child node of the DOM node has HTML tag <HR>, then divide this DOM node
<b>Rule 7</b>	If the background colour of this node is different from one of its children's, divide this node and at the same time, the child node with different background colour will not be divided in this round.
<b>Rule 8</b>	If the node has at least one text node child or at least one virtual text node child and the node's relative size is smaller than a threshold, then the node cannot be divided
<b>Rule 9</b>	If the child of the node with maximum size are smaller than a threshold (relative size), do not divide this node.
<b>Rule 10</b>	If previous sibling node has not been divided, do not divide this node
<b>Rule 11</b>	Divide this node
<b>Rule 12</b>	Do not divide this node


Table 4 Heuristic rules in Block Extraction phase

For different DOM nodes with different HTML tags, we will apply different rules. We listed them in below table.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Inline Text Node	√	√	√	√	√	√		√	√		√	
<TABLE>	√	√	√				√		√			√
<TR>	√	√	√				√		√			√
<TD>	√	√	√	√				√	√	√		√
<P>	√	√	√	√	√	√		√	√		√	
Other Tags	√	√	√	√		√		√	√		√	


Table 5 Different rules for different DOM nodes.

Let us consider an example shown in Figure 11. At the first round of block extraction, VB1, VB2\_1, VB2\_2, VB2\_3, VB3 and VB4 will be extracted and put into the pool. Below we explain how the VB2\_1, VB2\_2 and VB2\_3 are extracted in details. Figure 20 is a table, which is a part of the whole web page. Its DOM tree structure is shown on the left side. In the block



[Auctions Home](#) - [Yahoo! - Help](#)

Four ways to shop: [Shopping](#) [Used](#) [Auctions](#) [Classifieds](#)

powered by 

Welcome, loveixiaoyuer
 

[Submit Item](#) - [My Auctions](#) - [Options](#) - [Sign Out](#)

# Books & Comics

[Auctions](#) > [Books & Comics](#)

[Books & Comics](#)
[Advanced Search](#)

Categories

[Antique & Rare](#)  
[Audio & Large Print](#)  
[Bestsellers](#)  
[Children & Young Adult](#)  
[Classic Titles](#)  
[Comics](#)  
[Cooking, Food & Wine](#)  
[First Editions](#)  
[Foreign Language](#)  
[Lesbian, Gay & Bisexual](#)  
[Literature & Fiction](#)  
[Magazines](#)  
[Non-Fiction](#)  
[Poetry](#)  
[Self Help](#)  
[Travel, Reference & Education](#)  
[Other](#)


Superheroes



[Spider-Man](#)  

- [Superman, Batman, Incredible Hulk](#)
- [X-Men, Green Lantern, Avengers](#)
- [Captain America, Fantastic Four](#)
- [Flash Gordon, Wonder Woman](#)

Star Wars




[Darth Vader](#)  

- [Star Wars, Empire Strikes Back](#)
- [Return of the Jedi, Phantom Menace](#)
- [Attack of the Clones](#)
- [Obi-Wan Kenobi, Luke, Yoda, R2D2](#)
- [Darth Maul, Queen Amidala, Anakin](#)


Featured Auctions [View all...](#)

Promote your auctions here for a daily fee. [Learn more.](#)



Premium Sellers


- [Casey Creations](#)  
Children's Books & Records
- [My Book House](#)  
Rare, Unique and Hard to find books
- [Comics & Cards Unlimited](#)  
Comics & more - CGC...
- [InCOMICS Auction Booth](#)  
Comic Books and more! Oh yeah!




Inside Yahoo!

Fulfill all of your book & comics needs  
[Ask Yahoo! - Literature](#) - can't remember the name of an author? Ask Yahoo!  
  
[Books Shopping](#) - buy new books on Yahoo! Shopping  
  
[Ebooks Shopping](#) - check out new ebooks on Yahoo! Shopping  
  
[Yahoo! Book Clubs](#) - find out what other Yahoo! users are reading  
  
[Yahoo! Message Boards](#) - see what others are saying about books  
  
[Yahoo! News](#) - get all of the latest literary news  
  
[Oprah's Picks](#) - from the official Oprah Book Club site

Or find in:

 [Yahoo! Shopping](#)  
 Brand Names & Boutiques

 [Yahoo! Classifieds](#)  
 Your Local Listings

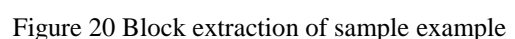
Popular Searches

[Harry Potter](#)  
[Stephen King](#)  
[Spider-Man](#)  
[Superman](#)  
[Rolling Stone](#)

[Auctions Home](#)



[Books & Comics](#)



### 5.2.2 Separator Detection

After all blocks are extracted, they are put into a pool for visual separator detection. Separators are horizontal or vertical lines in a web page that visually cross with no blocks in the pool. From a visual perspective, separators are good indicators for discriminating different semantics within the page. A visual separator is represented by a 2-tuple: (Ps, Pe), where Ps is the start pixel and Pe is the end pixel. The width of the separator is calculated by the difference between these two values.

Take Figure 21 as an example in which the black blocks represent the visual blocks in the page. For simplicity we only show the process to detect the horizontal separators. At first we have only one separator that is the whole pool. As shown in Figure 21 when we put the first block into the pool, it splits the separator into S1 and S2. It is the same with the second and third block. When the fourth block is put into the pool, it crosses the separator S2 and covers the separator S3, the parameter of S2 is updated and S3 is removed. At the end of this process, the two separators S1 and S3 that stand at the border of the pool are removed.

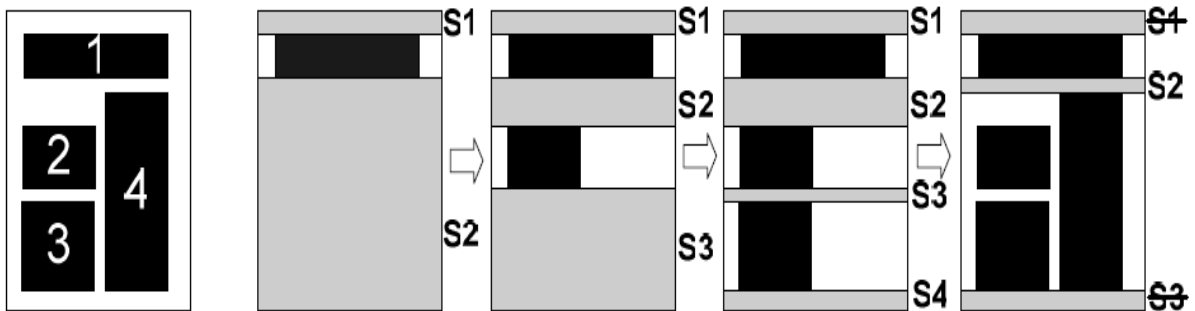


Figure 21 Separator Detection Process

The separators are used to distinguish blocks with different semantics, so the weight of a Separator can be assigned based on the visual difference between its neighbouring blocks. The following rules are used to set a weight to each separator:

- The greater the distance between blocks on different side of the separator, the higher then weight.
- If a visual separator is overlapped with some certain HTML tags (e.g., the `<HR>` HTML tag), its weight is set to be higher.
- If background colours of the blocks on two sides of the separator are different, the weight will be increased.

- For horizontal separators, if the differences of font properties such as font size and font weight are bigger on two sides of the separator, the weight will be increased. Moreover, the weight will be increased if the font size of the block above the separator is smaller than the font size of the block below the separator.
- For horizontal separators, when the structures of the blocks on the two sides of the separator are very similar (e.g. both are text), the weight of the separator will be decreased.

### 5.2.3 Content Structure Construction

When separators are detected and separators' weights are set, the content structure can be constructed accordingly. The construction process starts from the separators with the lowest weight and the blocks beside these separators are merged to form new blocks. This merging process iterates till separators with maximum weights are met. The DoC of each new block is set based on the maximum weight of the separators in the block's region. After that, each leaf node is checked whether it meets the granularity requirement. For every node that fails, we go to the Visual Block Extraction step again to further construct the sub content structure within that node. If all the nodes meet the requirement, the iterative process is then stopped and the vision-based content structure for the whole page is obtained. The common requirement for DoC is that  $\text{DoC} > \text{PDoC}$ , if PDoC is pre-defined.

In summary, the introduced DDE algorithm takes advantage of visual cues to obtain the Table-based content structure of a web page and thus successfully bridges the gap between the DOM structure and the semantic structure. The page is partitioned based on visual separators and structured as a hierarchy. This semantic hierarchy is consistent with human perception to some extent. DDE is also very efficient. Since we trace down the DOM structure internally for visual block extraction and do not analyse every basic DOM node, the algorithm is totally top-down.

### 5.3 Noise Removal from Web Page using DDE Algorithm

One crucial step in web data mining, including structured extraction, is the cleaning phase that takes place before extracting the information. One cannot expect to get good results in the extraction phase without cleaning and removing the undesired noise first. [55] Mention that despite the importance of this task, relatively some work has been done in this area and, while reviewing up to date related work, we still have the impression that this is an underdeveloped field. In structured extraction, most of the existing approaches use some sort of the pattern recognition to identify the records [56] present in the page. The problem is that, usually, we are interested only in the main content region but other regions of the page (menus, ad, extra, etc.) often contain repeating pattern that are outputted as noise results. So it is useful to clean up a web page before extracting the records from it.

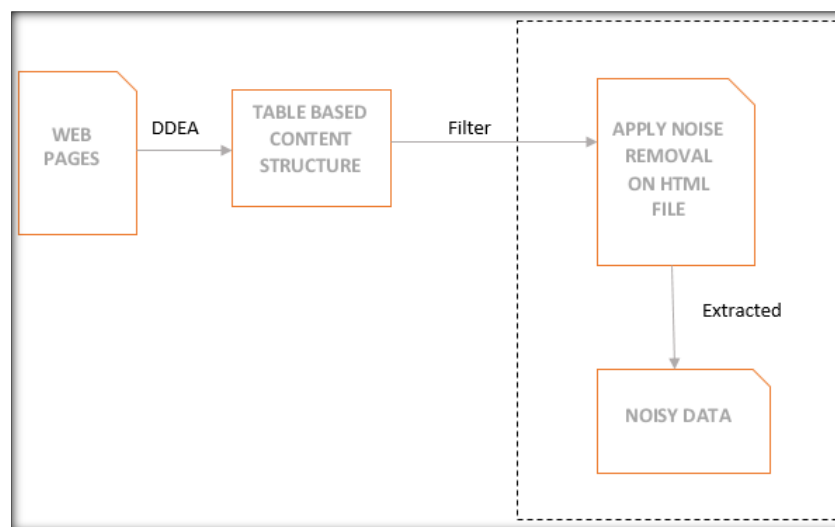


Figure 22 Process flow of noise removal

The given DDE Algorithm extract the content structure from the Web page and Remove the Noisy data from the extracted dynamic data of Web page, which helps to remove the unwanted data and redundant data.

In this sub section we using some algorithm to remove the noise from Web page.



- **Tag\_PathSequence\_Filter ( ):** It is the main algorithm, which receives a HTML file as input and return a pruned DOM tree with the main content region.
- **Convert\_Tree\_ToSequence ( ):** It converts the Web page DOM tree into a tag path sequence.
- **Search\_Region ( ):** It is the actual search for the main region of the TPS.
- **Filter\_Alphabet ( ):** It filter an alphabet, removing lower frequency symbols, making the overall algorithm more robust and resistant to noise.
- **Prune\_DOMTree ( ):** It prunes the original DOM tree, leaving only the main content region reported by Search\_Region, keeping the original structure of the document.

The complexity of the above algorithms are  $O(n)$ .

## 5.4 Performance Evaluation of DDE Algorithm

In this section we evaluate the performance on the bases of the number of segments in the Web page. Here we compare Dynamic Data Extraction Algorithm with the three similar algorithms, which are FULLDOC, DOMPS, and VIPS. We can say that Dynamic Data Extraction Algorithm is the extended form of VIPS algorithm because DDE algorithm parse the DOM tree and stored the extracted segmented data in the form of table. Using DDE algorithm, we remove the noisy data as well as extract the unwanted tags and unwanted data. We compare these four algorithms with Average Precision percentage.

$$Precision = \frac{\text{total number of corrected data extracted}}{\text{total number of data extracted}}$$

Calculate the precision of different segments of the Web page by using fulldoc, domps, vips and dynamic data extraction algorithm. The performance is evaluated on the bases of the threshold (baseline) which is 16.55 []. If the precision percentage is lying below the threshold level than algorithm has the worst performance and if precision percentage is lying above the threshold level then the performance of the algorithm is good.

Number of Segments	Baseline	FULLDOC	DOMPS	VIPS	DDE
3		17.56	17.94	18.01	18.23
5		17.46	18.15	19.39	19.86
10		19.1	18.05	19.92	20.01
20	16.55	17.89	19.24	20.98	20.74
30		17.4	19.32	19.68	20.32
40		15.5	19.57	17.24	19.68
50		13.82	19.67	16.63	20.04
60		14.4	18.58	16.37	19.83

Figure 23 Performance comparison of the DDE Algorithm

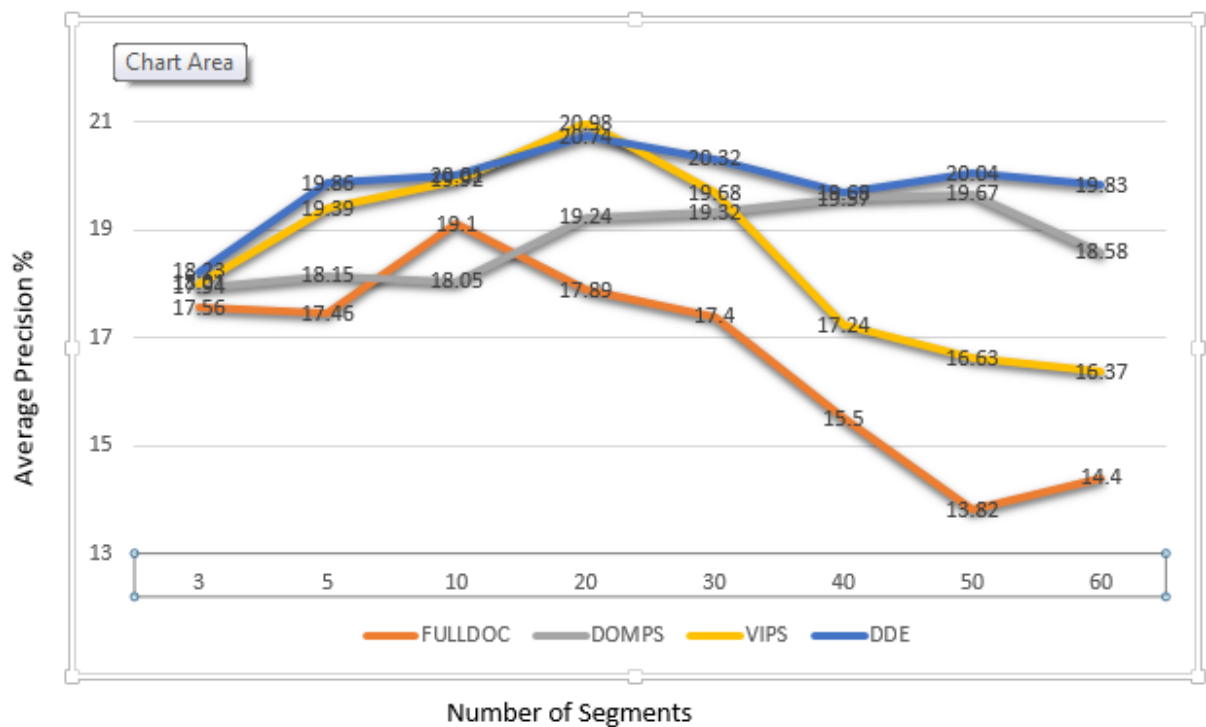


Figure 24 Performance comparison through the graph

As can be seen, the average retrieval precision can be improved after partitioning pages into blocks, no matter which segmentation algorithm is used. In the case of FULLDOC, the maximal average precision is 19.10% when the number of segments of the Web page is 10. DOMPS obtains 19.67% when 50 segments of blocks are used, a little better than FULLDOC. VIPS gets the result 20.98% when the 20 segments are used and DDE algorithm gets best result when 20 segments are used. But if we see the overall performance of the algorithm then DDE

algorithm shows the best result over all three algorithm. DDE algorithm shows precision percentage above than the threshold level in all number of segment. So we can say that Dynamic Data Extraction Algorithm is good among all the other three algorithms.

## 5.5 Result of DDE Algorithm

DDE Algorithm has a following steps:

1. Run the file of DDE algorithm, it will display an output window.

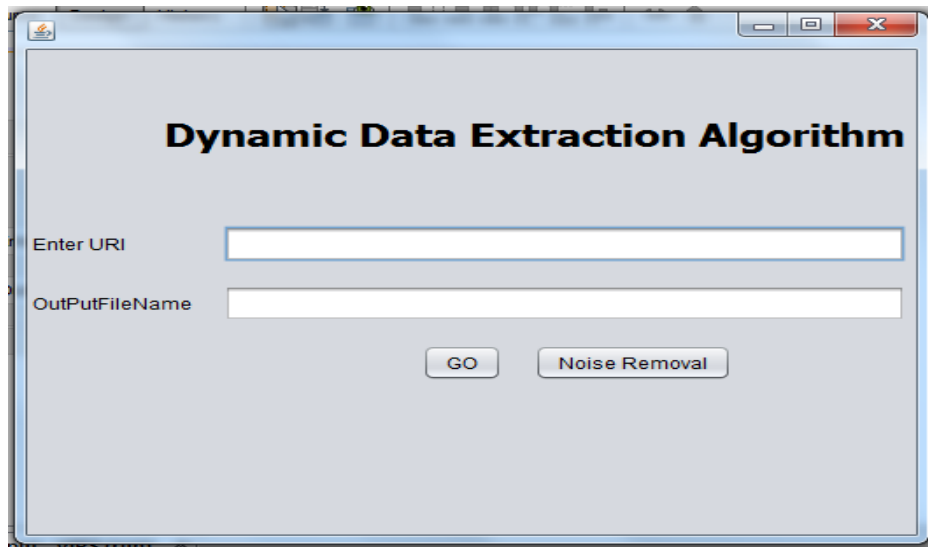


Figure 25 Output window of DDEA

2. Enter the URL of the Website and output file name in text area then click on GO button.

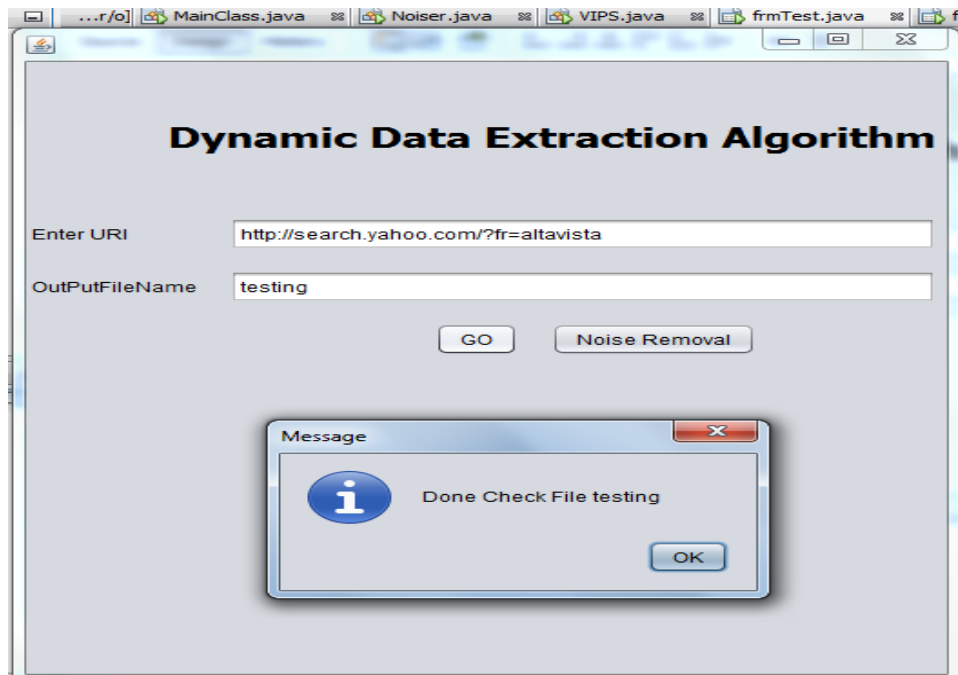


Figure 26 Result after sorted the output file

After extracted structure of Web page Content, it will show the message box that output file is stored and Click OK.

3. Go to the pre-defined path where output file is stored and open the file which contain content structure of Web page in the form of table.

**Table 1**

Row Number		
1	Home	<a href="#">Link&lt;&lt;Home</a>
2	Mail	<a href="#">Link&lt;&lt;Mail</a>
3	News	<a href="#">Link&lt;&lt;News</a>
4	Sports	<a href="#">Link&lt;&lt;Sports</a>
5	Finance	<a href="#">Link&lt;&lt;Finance</a>
6	Weather	<a href="#">Link&lt;&lt;Weather</a>
7	Games	<a href="#">Link&lt;&lt;Games</a>
8	Groups	<a href="#">Link&lt;&lt;Groups</a>
9	Answers	<a href="#">Link&lt;&lt;Answers</a>
10	Screen	<a href="#">Link&lt;&lt;Screen</a>
11	Flickr	<a href="#">Link&lt;&lt;Flickr</a>
12	Mobile	<a href="#">Link&lt;&lt;Mobile</a>





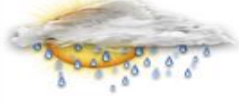
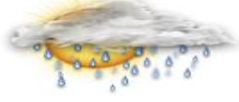
**Table 2**

Row Number		
1	Celebrity	<a href="#">Link&lt;&lt;Celebrity</a>
2	Shine	<a href="#">Link&lt;&lt;Shine</a>
3	Movies	<a href="#">Link&lt;&lt;Movies</a>
4	Music	<a href="#">Link&lt;&lt;Music</a>
5	TV	<a href="#">Link&lt;&lt;TV</a>
6	Health	<a href="#">Link&lt;&lt;Health</a>
7	Beauty	<a href="#">Link&lt;&lt;Beauty</a>
8	Food	<a href="#">Link&lt;&lt;Food</a>
9	Tech	<a href="#">Link&lt;&lt;Tech</a>
10	Shopping	<a href="#">Link&lt;&lt;Shopping</a>
11	Travel	<a href="#">Link&lt;&lt;Travel</a>
12	Autos	<a href="#">Link&lt;&lt;Autos</a>
13	Homes	<a href="#">Link&lt;&lt;Homes</a>

**Table 3**

Row Number		
1	Palestinians flee north Gaza after Israel...	<a href="#">Link&lt;&lt;Palestinians flee north Gaza after Israel...</a>
2	Iraq parliament postpones decision on new...	<a href="#">Link&lt;&lt;Iraq parliament postpones decision on new...</a>
3	Islamist militia attacks rivals at main Libya...	<a href="#">Link&lt;&lt;Islamist militia attacks rivals at main Libya...</a>
4	AP Analysis: Mideast crisis a strategic...	<a href="#">Link&lt;&lt;AP Analysis: Mideast crisis a strategic...</a>
5	Obama may hold fix to flood of immigrant kids	<a href="#">Link&lt;&lt;Obama may hold fix to flood of immigrant kids</a>

**Table 4**

Row Number					
1			Today	77Å°	67Å°
2			Tomorrow	79Å°	68Å°
3			Tuesday	79Å°	67Å°

**Table 5**

Row Number										
1	Ariana Grande	<a href="#">Link&lt;&lt;Ariana Grande</a>	Brenda Song	<a href="#">Link&lt;&lt;Brenda Song</a>	Stacy Keibler	<a href="#">Link&lt;&lt;Stacy Keibler</a>	Kate Upton	<a href="#">Link&lt;&lt;Kate Upton</a>	Amber Alert	<a href="#">Link&lt;&lt;Amber Alert</a>
2	Cheryl Burke	<a href="#">Link&lt;&lt;Cheryl Burke</a>	Sierra LaMar	<a href="#">Link&lt;&lt;Sierra LaMar</a>	Laura Prepon	<a href="#">Link&lt;&lt;Laura Prepon</a>	Migraine...	<a href="#">Link&lt;&lt;Migraine...</a>	Halle Berry	<a href="#">Link&lt;&lt;Halle Berry</a>

**Table 6**

Row Number		
1	Yahoo	
2	Help	<a href="#">Link&lt;&lt;Help</a>
3	Privacy	<a href="#">Link&lt;&lt;Privacy</a>
4	Terms	<a href="#">Link&lt;&lt;Terms</a>
5	Advertise	<a href="#">Link&lt;&lt;Advertise</a>
6	Submit Your Site	<a href="#">Link&lt;&lt;Submit Your Site</a>

Figure 27 Result of extracted Dynamic data

- After extracted dynamic data, now we remove the noisy data from the extracted data.  
Click on noise removal button and input the html file extracted data and Click on submit button.

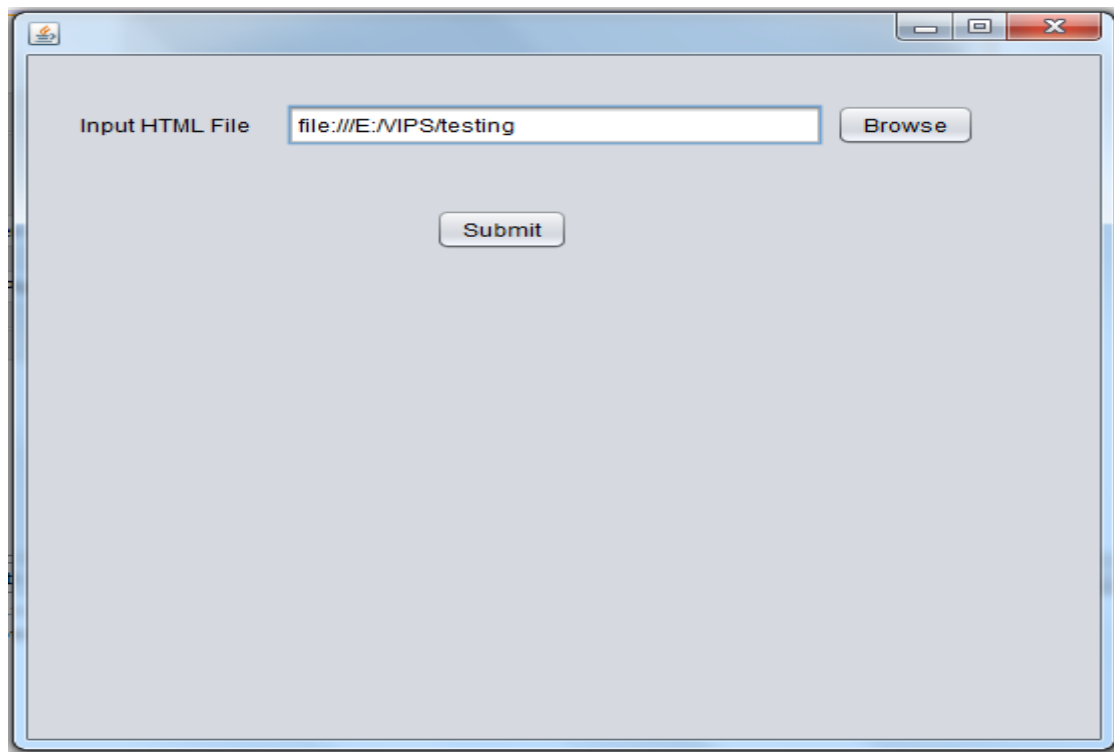


Figure 28 Input the file for noise removal

5. Open the file from pre-defined path which contain the file of extracted noisy data.

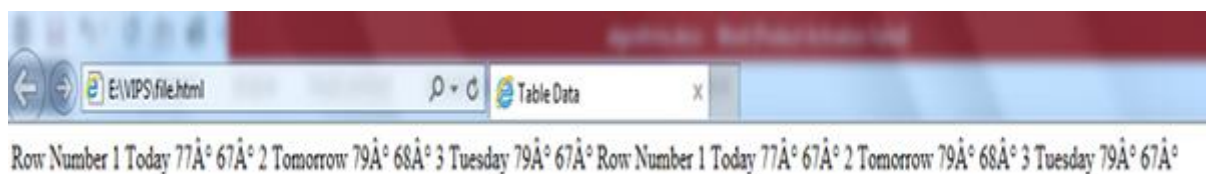


Figure 29 Display the result Noisy Data

## 5.6 Empirical Results

In this section we have done empirical study regarding size of the Web page and Noisy data extracted from the DDE algorithm. With the help of DDE algorithm we observe that noise is the directly proportional to the Web page size.

To prove this we collect the high page rank Web site from <http://doheth.co.uk/info/list-of-websites-with-high-page-rank.php>. Page Rank (PR) is the number of points out of ten that signifies the importance of a site to Google. The PageRank number appears to be logarithmic, with a single point representing an order of magnitude difference of importance. After that apply DDE Algorithm on Web pages, then calculate the Dynamic Web page size and noisy data size of Dynamic Web page, then plot the graph between Web page size and Extracted Noisy data size.

S.no	Web Page Size (KB)	Extracted Noisy Data Size (KB)
1	8	1
2	9	2.1
3	10	1.5
4	12	1
5	13	3
6	14	3
7	16	4.1
8	18	2
9	19	4
10	22	2.3
11	23	4
12	24	6
13	25	5.5
14	27	6
15	28	7
16	28	4.8
17	31	6
18	35	7
19	38	6
20	40	6.8
21	41	9.6
22	63	9.5
23	82	15
24	83	16.4

Figure 30 Web page size and extracted noisy data size

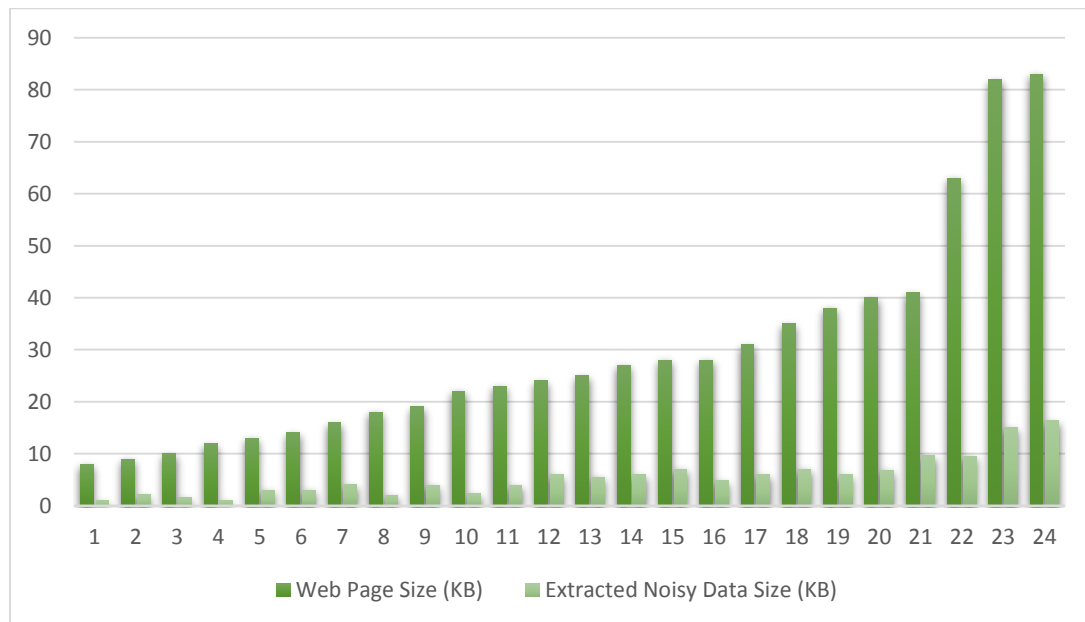


Figure 31 Graphical result of Web page size and extracted noisy data size

## **5.7 Cost Constructive Model in Web Application**

In this section we use Constructive Cost Model in Web application to calculate the development time of the web application through which we can calculate the maintenance cost of the Web pages. Constructive Cost Model is one of the cost estimation models which contribute three different models namely basic, intermediate and detailed model. Maintenance of software is very much important so as to increase the functionality of software and decrease the cost incurred in extracting new system. The first level of COCOMO is Basic which is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes. In the Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases. Here, in this paper, we mainly estimate maintenance cost of software using basic and intermediate model by maintaining Annual change traffic and thousand source lines of code as constant. We examine these values by varying on each case, thereby obtaining the final maintenance costs in man-months.

### **5.7.1 Calculation of Web Size Metrics**

The size metric used in other metrics like productivity metric and cost estimation metric. One of the most common metric to calculate the size of the software is Lines of Code shortly termed as LOC but in Web page we are using Number of Tags (NoT) to calculate the size. Now the problem is to calculate the number of tags in the dynamic data of the Web pages because we are working here all about on the dynamic data of the Web page. We already introduced our Dynamic Data Extraction Algorithm which helps us to extract the dynamic data from the Web page.

From there we extract the dynamic data and then calculate the number of tags of Web page. The NoT in the Web pages are the number of tags used in the html file. So after getting dynamic data of Web page, we extract number of tag used in the html file of the dynamic data. For extraction of the html tags from file, we make a tool through which we can calculate the number of tags.

The size metric is measured in terms of kilo lines of code (KLOC) and lines of code (LOC). Here code is the all executable line as well as non-executable line, declaration, header and all tags etc.



```
Command Window

>> [ndata, tdata]=xlsread('Book1.xlsx','Sheet1');

result=zeros(10,1,'int32');
m=1;
for i=1:353
    for j=1:12
        if (strcmpi(tdata(i,j),'a'))
            result(m,1)=1;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'html'))
            result(m,1)=2;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'title'))
            result(m,1)=3;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'head'))
            result(m,1)=4;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'body'))
            result(m,1)=5;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'table'))
            result(m,1)=6;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'tr'))
            result(m,1)=7;
            m=m+1;
```

```
Command Window

        elseif (strcmpi(tdata(i,j),'td'))
            result(m,1)=8;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'tc'))
            result(m,1)=9;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'th'))
            result(m,1)=10;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'p'))
            result(m,1)=11;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h1'))
            result(m,1)=12;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h2'))
            result(m,1)=13;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h3'))
            result(m,1)=14;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h4'))
            result(m,1)=15;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h5'))
            result(m,1)=16;
            m=m+1;
        elseif (strcmpi(tdata(i,j),'h6'))
```

```

result(m,1)=59;
m=m+1;
elseif (strcmpi(tdata(i,j),'ins'))
    result(m,1)=60;
    m=m+1;
elseif (strcmpi(tdata(i,j),'img'))
    result(m,1)=61;
    m=m+1;
elseif (strcmpi(tdata(i,j),'i'))
    result(m,1)=62;
    m=m+1;
elseif (strcmpi(tdata(i,j),'div'))
    result(m,1)=63;
    m=m+1;
end

end
end

count=0;
for i=1:3056
    if result(i,1)==0
        count=count+1;
    end
end
end

```

Figure 32 Snapshot of the Mat lab Program

### 5.7.2 Calculation of Effort, Development Time and Maintenance Cost

To estimate the Effort (Development Cost) and Development Time, collect Web pages data with different LOC. Then categorize different size of LOC into three modes of COCOMO are Organic, Semi-detached and Embedded.

	Size	Innovation	Deadline	Dev. Environment
<b>ORGANIC</b>	Small	Little	Not Tight	Stable
<b>SEMI-DITACHED</b>	Medium	Medium	Medium	Medium
<b>EMBEDDED</b>	Large	Greater	Tight	Complex Hardware

Figure 33 Three modes of COCOMO model

Here we assume that,

1. If the number of NoT is in between 2-600 then it belong to the Organic mode of the COCOMO
2. If the number of NoT is in between 600-1000 then it belongs to the Semi-Detached mode.
3. If the number of NoT is in between above 1000 then it belongs to the Embedded mode.

There are COCOMO coefficient for all the modes.

Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Figure 34 COCOMO Coefficient

### 5.7.2.1 Organic Mode

- a) First we collect the small number of NoT of dynamic data extracted from Web pages which does not contain noise.

$$\begin{aligned} \text{Effort (Development Cost)} &= a (\text{NoT})^b \\ \text{Development Time} &= c (E)^d \end{aligned}$$

To estimate the maintenance cost, another parameter is needed: the annual change traffic (ACT) which consists of the proportion of original instructions that undergo a change during a year by addition or modification.

$$\text{ACT} = (\text{NNL} + \text{NML}) / \text{NOL}$$

Where,

NNL is the number of new lines

NML is the number of modified lines

NOL is the number of Total original lines

We assume that number of new lines are 500, modified lines are 2500 and total original lines are 10,000.

$$\text{ACT} = (2500 + 500) / 10,000 = 0.3$$

$$\text{Maintenance Cost} = \text{ACT} * \text{Development}$$

NoT	Development Cost	Development Time	Maintenance Cost
194	605	28.5	181
258	817	32	245
302	964	34	289
344	1105	36	331
361	1163	37	348
378	1220	37	336
408	1322	38	396
437	1421	40	426
445	1448	40	434
453	1476	41	442
455	1482	41	444
542	1782	43	534
551	1813	43	543
552	1816	43	544
553	1820	44	546
599	1979	45	593

Figure 35 Calculation of development cost, development time and maintenance cost

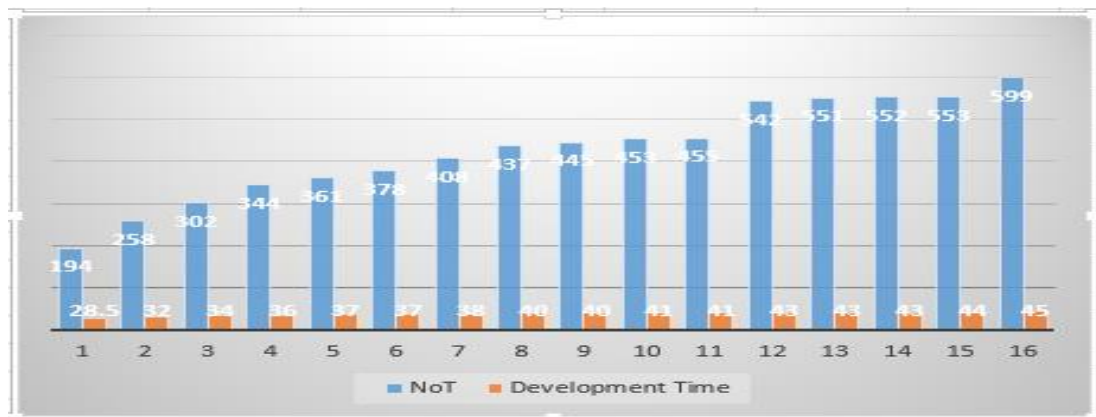


Figure 36 Graphical result of development time and NoT

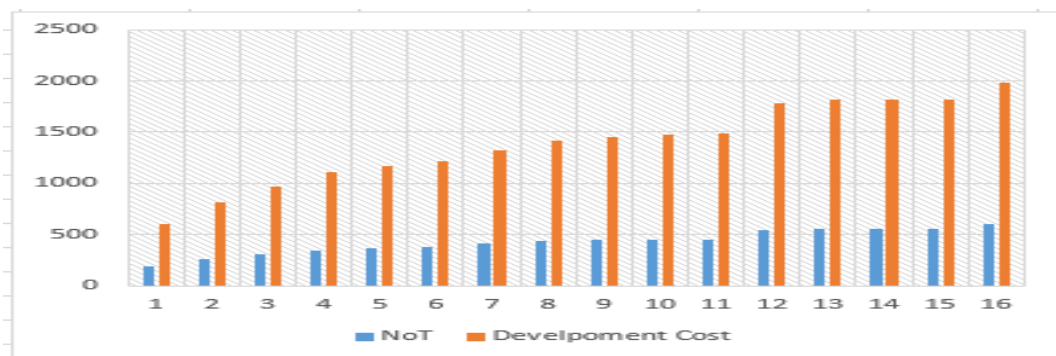


Figure 37 Graphical result development cost and NoT

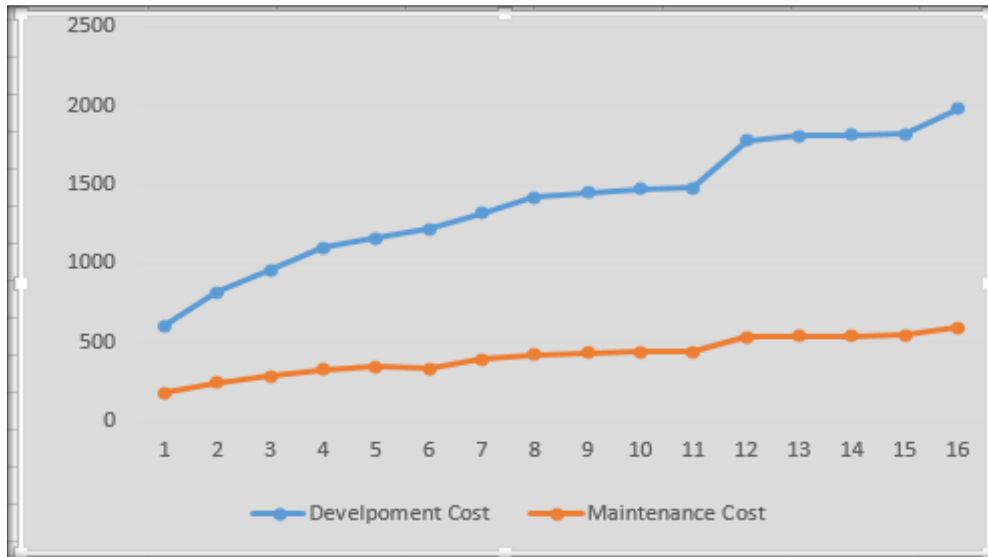


Figure 38 Graphical result of maintenance cost and development cost

- a) First we collect the small number of NoT of dynamic data extracted from Web pages which contain noise.

$$\begin{aligned} \text{Effort (Development Cost)} &= a (\text{NoT})^b \\ \text{Development Time} &= c (E)^d \\ \text{Maintenance Cost} &= \text{ACT} * \text{Development Cost} \end{aligned}$$

$$\text{ACT} = 0.3$$

NoT	Development Cost	Maintenance Cost
201	629	353
278	884	380
322	1031	430
358	1152	465
381	1230	491
398	1288	501
430	1083	520
440	1431	534
468	1527	553
478	1561	691
487	1592	701
567	1868	854
580	1913	890
592	1955	629
599	1979	901
619	2048	921

Figure 39 calculation of metrics for Noisy Web page

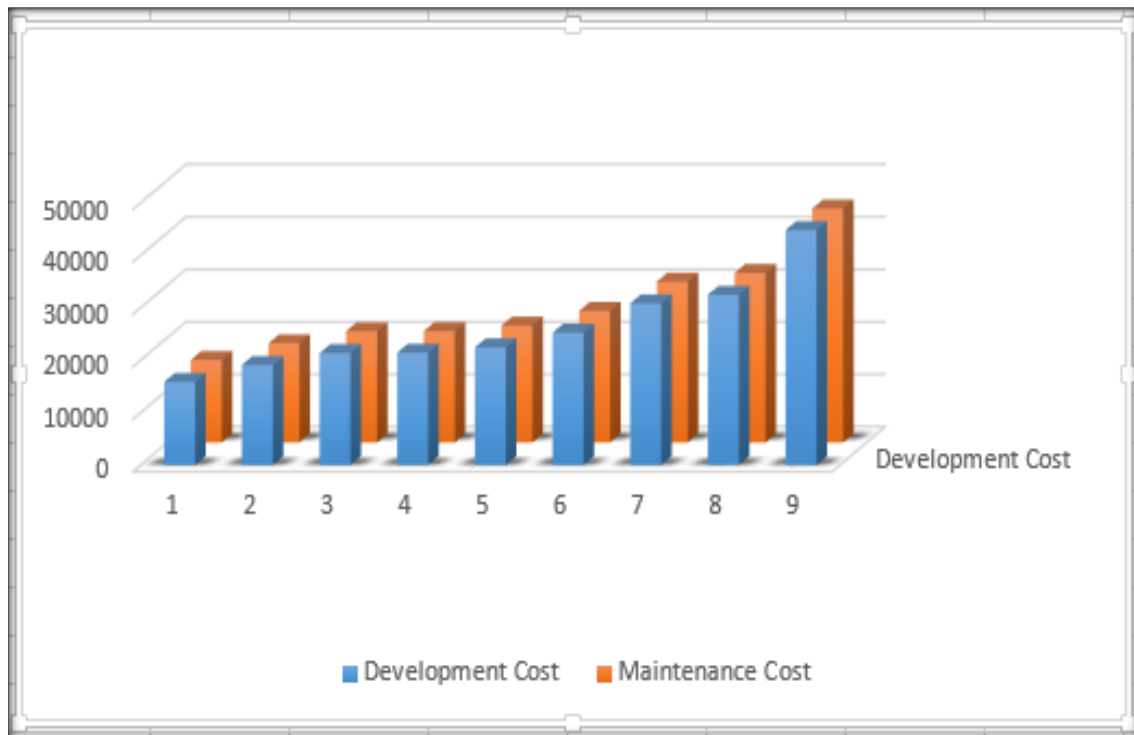


Figure 40 Graphical result of development cost and maintenance cost for Noisy Web page

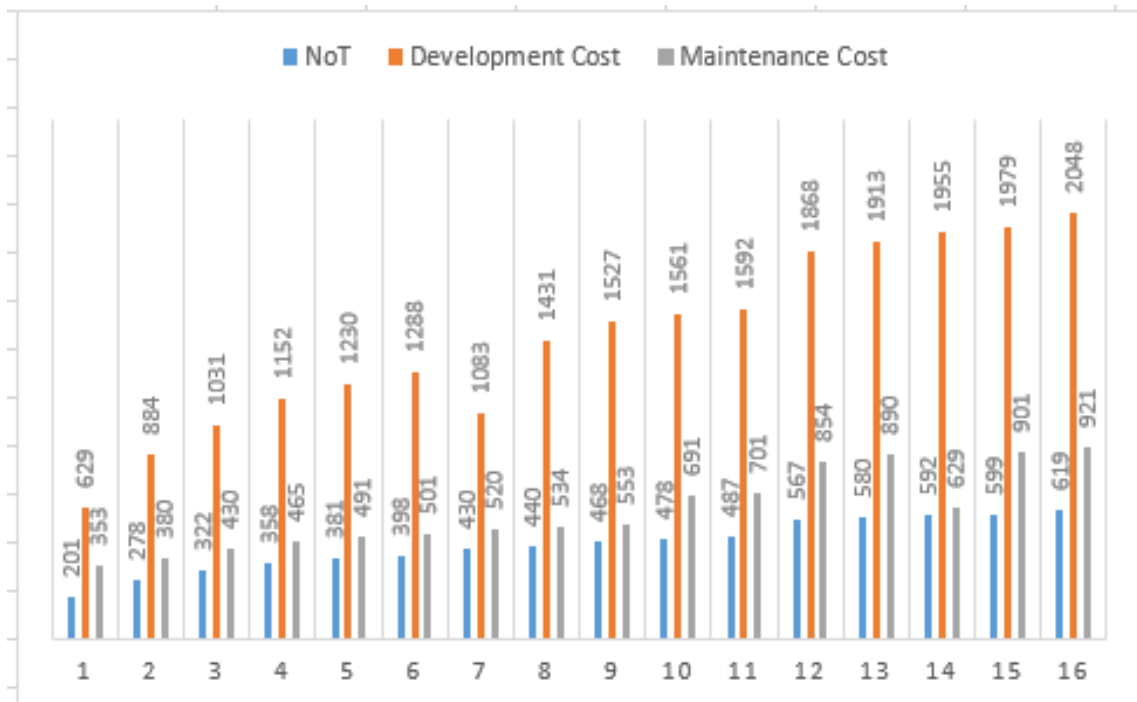


Figure 41 graphical result of NoT, development cost and maintenance cost

### 5.7.2.2 Semi-Detached Mode

- a) First we collect the medium number of NoT of dynamic data extracted from Web pages which does not contain noise.

$$\text{Effort (Development Cost)} = a (\text{NoT})^b$$

$$\text{Development Time} = c (E)^d$$

$$\text{Maintenance Cost} = \text{ACT} * \text{Development Cost}$$

Where ACT = 0.6.

NoT	Development Cost	Development Time	Maintenance Cost
639	2647	39.4	794
655	2717	39.7	815
657	2726	39.8	818
663	2752	39.9	826
708	2948	40.9	884
730	3045	41.4	914
762	3185	42	956
788	3299	42.5	990
840	3528	43.6	1058
857	3577	43.8	1073
898	3784	44.6	1135
913	3851	44.9	1155
986	4175	46.2	1253
991	4197	46.3	1259
997	4224	46.4	1267
1000	4237	46.5	1271

Figure 42 Calculation of metrics for semi-detached



Figure 43 Graphical result of NoT, development cost and maintenance cost

- b) First we collect the medium number of NoT of dynamic data extracted from Web pages which contain noise.

NoT	Development Cost	Maintenance Cost
692	4550	2348
708	4668	2409
710	4683	2416
720	4757	2453
761	5061	2605
787	5255	2702
815	5465	2806
845	5691	2916
897	6084	3117
910	6183	3162
955	6527	3338
968	6626	3391
1048	2414	1282
1062	7351	3746
1078	7475	3808
1098	7576	3989

Figure 44 Calculation of metrics for noisy web page

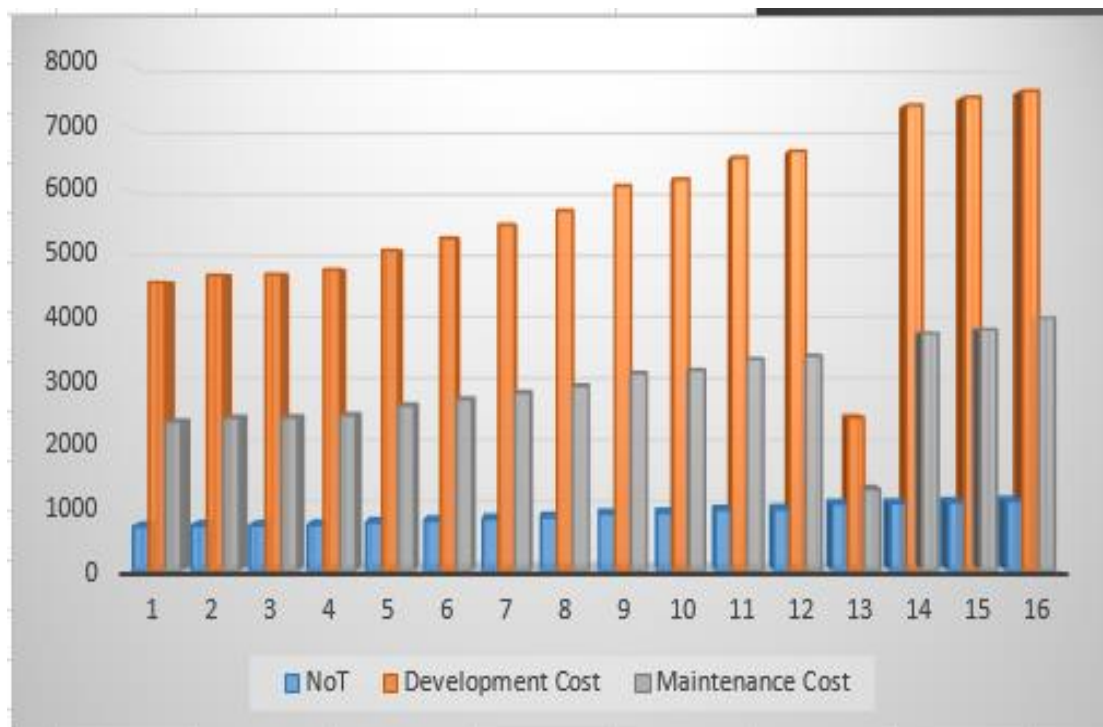


Figure 45 Graphical result for noisy Web page



### 5.7.2.3 Embedded Mode

- a) First we collect the large number of NoT of dynamic data extracted from Web pages which does not contain noise.

$$\text{Effort (Development Cost)} = a (\text{NoT})^b$$

$$\text{Development Time} = c (E)^d$$

$$\text{Maintenance Cost} = \text{ACT} * \text{Development Cost}$$

Where ACT = 0.9

NoT	Development Cost	Development Time	Maintenance Cost
1005	14417	53.5	4325
1206	17944	57.4	5383
1344	20435	59.8	6131
1358	20691	60.1	6207
1400	21461	60.8	6438
1584	24889	63.7	7467
1876	30491	68	9147
1925	31440	68.7	9432
2491	42850	75.8	12855

Figure 46 calculation of metrics for Embedded Mode

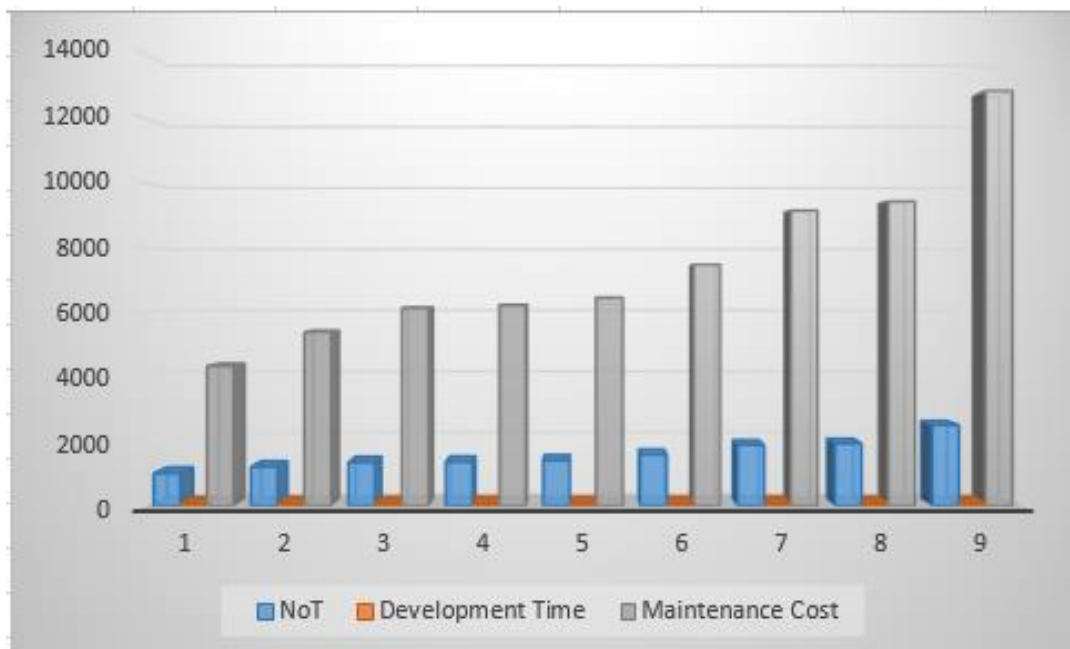


Figure 47 Graphical results for NoT, development time and maintenance cost

- b) First we collect the large number of NoT of dynamic data extracted from Web pages which contain noise.

NoT	Development Cost	Maintenance Cost
1087	15840	15690
1267	19038	18898
1392	21314	21164
1396	21387	21237
1452	22421	22271
1601	25209	25059
1898	30784	30634
1975	32432	32284
2582	44735	44585

Figure 48 Calculation of metrics for noisy Web page

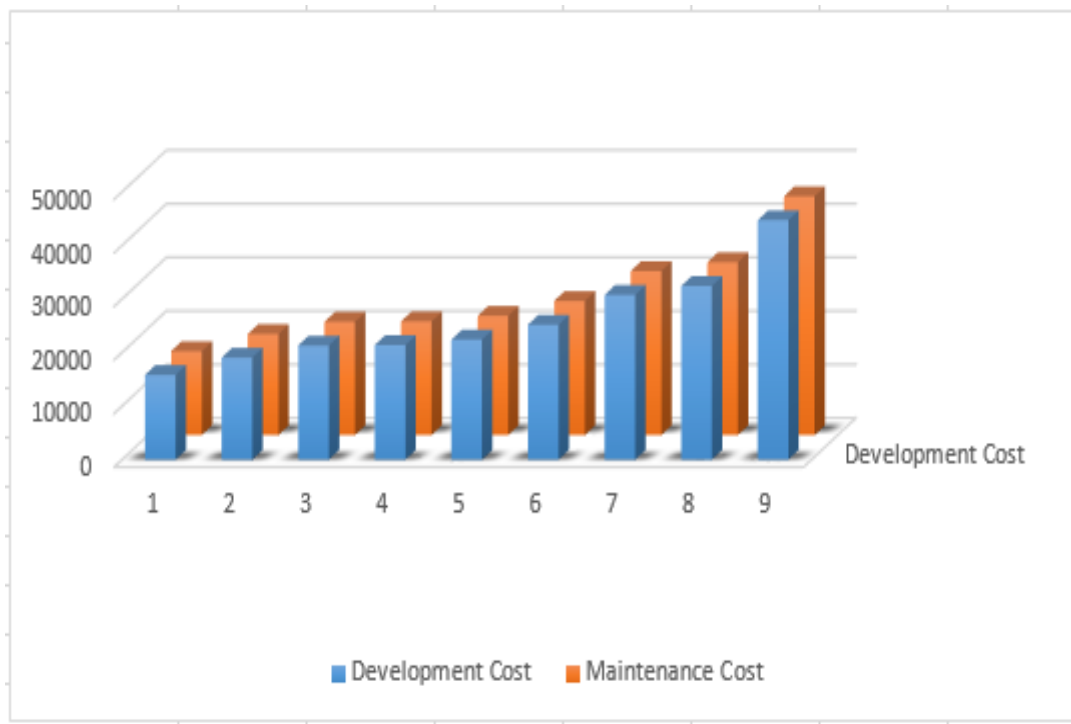


Figure 49 Graphical result of development cost and maintenance cost

### 5.7.3 Compare the Maintenance Cost

This subsection compare the Maintenance cost of noisy lines of code and without noisy lines of code. We can see the difference in the maintenance cost, achieve good result by using dynamic data extraction algorithm. Here we compare in all three modes of Cost Constructive Model.

#### a) Organic Mode

S.No	Maintenance Cost (WN)	Maintenance Cost (N)
1	181	353
2	245	380
3	289	430
4	331	465
5	348	491
6	336	501
7	396	520
8	426	534
9	434	553
10	442	691
11	444	701
12	534	854
13	543	890
14	544	629
15	546	901
16	593	921

Figure 50 Comparison of maintenance cost for organic mode



Figure 51 Graphical result of comparison of maintenance cost

Where,

WN is the without noise, N is with noise

b) Semi-Detached Mode

S.no	Maintenance Cost (WN)	Maintenance Cost (N)
1	794	2348
2	815	2409
3	818	2416
4	826	2453
5	884	2605
6	914	2702
7	956	2806
8	990	2916
9	1058	3117
10	1073	3162
11	1135	3338
12	1155	3391
13	1253	1282
14	1259	3746
15	1267	3808
16	1271	3989

Figure 52 Comparison of maintenance cost for semi-detached mode

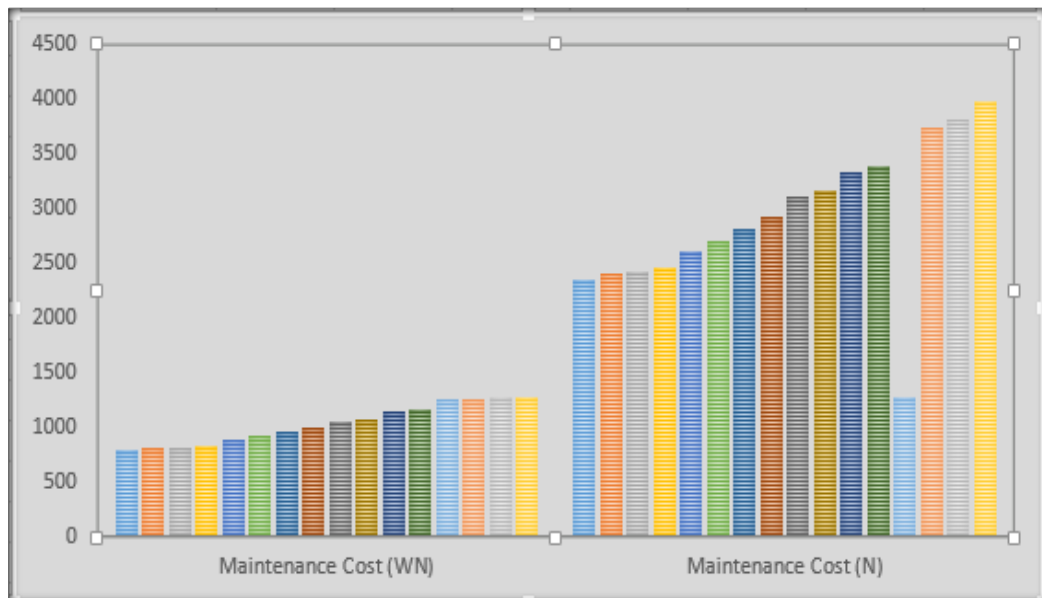


Figure 53 Graphical result of maintenance cost

Here we observe that Maintenance Cost (WN) is almost half of the Maintenance Cost (N).

c) Embedded Mode

S.no	Maintenance Cost (WN)	Maintenance Cost (N)
1	4325	15690
2	5383	18898
3	6131	21164
4	6207	21237
5	6438	22271
6	7467	25059
7	9147	30634
8	9432	32284
9	12855	44585

Figure 54 Comparison of maintenance cost for embedded mode

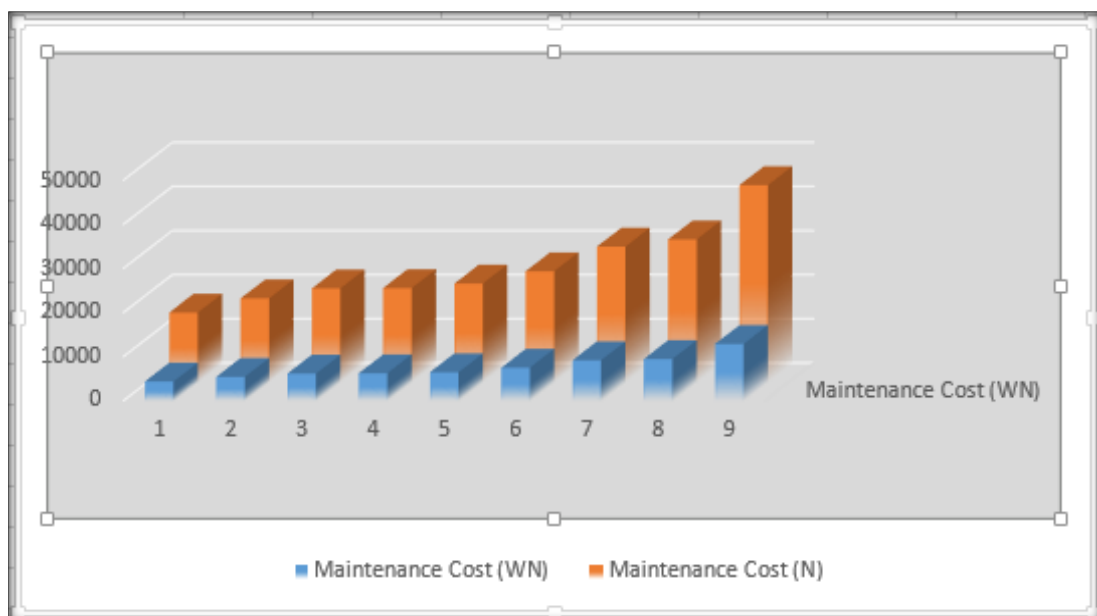


Figure 55 Graphical result of maintenance cost

We can see 20 percent maintenance cost will inclined in dynamic Web page with as compared to without noise for Organic mode, around 50 percent maintenance cost will inclined for Semi-detached mode and around 80 percent maintenance cost will inclined for Embedded mode. If Number of Tags increases, size of noisy data increases that will directly affect the maintenance cost.

## **5.8 Optimizing Web Consistency Using Visual Page Analysis**

Due to the growing importance of the World Wide Web, archiving it has become crucial for preserving useful source of information. To maintain a Web archive up-to-date, crawlers harvest to the web by iteratively downloading new version of document. However, it is frequent that crawler retrieve pages with unimportant changes such as advertisement which are continually updated. Hence, web archive system waste time and space for indexing and storing useless page version. An effective method is required to know accurately when and how often important changes between versions occur in order to efficiently archive web pages. Here we focuses on addressing this requirement through a new web archiving approach that detects important changes between page versions. This approach consist in archiving the visual layout structure of a web page represented by semantic blocks. This approach seeks to describe and to examine the various related issues such as using the importance of changes between versions to optimize web crawl scheduling.

We describe the steps of this approach which points out the problem that is detecting/ analyzing changes importance between page versions. Also, a crawl scheduling strategy which uses the analysis of change importance.

### **5.8.1 Visual Page Segmentation**

Dynamic Data Extraction Algorithm is used to segment a web page into nested semantic blocks based on suitable nodes in the HTML DOM tree of the page. It detects the horizontal and vertical separators in a web page. Based on those separators, it builds the semantic tree of the web page partitioned into multiple blocks. The root is the whole page. Each block is represented as a node in the tree as shown in Figure 56. To complete the semantic tree of the whole page, we extended the Dynamic Data Extraction algorithm by extracting links, images and text for each block. As illustrated in Figure 56, each block node has additional children nodes: Links, Images and Texts that gather respectively all hyperlinks, pictures and text contained in the block. All nodes of the page are uniquely identified by an ID attribute. This ID is a hash value computed using the node's content and its children nodes content: if matched nodes (nodes at the same position in two successive versions) have different ID values, then their content has been necessarily updated. Leaf nodes have other attributes such as the name and the address for the hyperlink. Our extended DDE algorithm generates, as output, a Vi-XML document that

describes the complete hierarchical structure of the web page. The structure of such a document is shown in Figure 56.

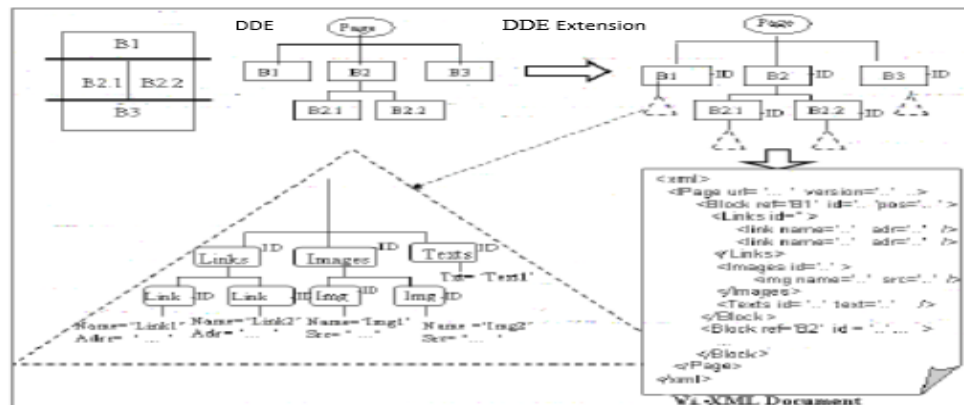


Figure 56 Extended DDE Algorithm

### 5.8.2 Change Detection

We describe the change detection algorithm because existing ones are generic-purpose. We would like to add some specific criteria for comparing attributes nodes, such as detecting updated links, if one of their attributes (name or address) is modified. We want also to detect an updated text in two matched blocks based on a textual similarity/distance score (e.g. number of shared words). Another specificity of our approach is that we need to detect changed elements inside a block and moved elements from one block to another, but detecting moved element inside a same block is useless because no information has been added or deleted in the block. Also, we would like to detect that the structure of the page at level of blocks is changed (inserted/deleted block, etc.) from one version to another. The Vi-DIFF algorithm detects two types of changes; structural and content changes. The structural changes (insert, delete and shift) typically modify the structure of XML document at level of blocks whereas content changes (insert, delete, update and move) modify the textual content at level of links, images and texts. All detected change operations are then described in a Vi-Delta file. If we assume that there is no change in structure then the complexity of Vi-DIFF is log linear  $O(n \cdot \log(n))$  where  $n$  is the total number of nodes. If there are structural changes, in the worst case (all the structure is changed), the complexity is quadratic  $O(n^2)$  but it is worth to notice that always remains small. The Vi-DIFF algorithm detects changes between two Vi-XML pages. We intend in future work to extend this algorithm in order to detect changes between two versions of sites

instead of two pages. The extended Vi-DIFF must produce, as output, a complete Vi-Delta describing changes between each page of the two site versions.

### 5.8.3 Changes Importance

Based on the Vi-Delta produced by Vi-DIFF, we explain a function that evaluates the importance of detected changes. This function depends on three major parameters:

- *Importance of updated block*: Typically, the most important information is on the center and the advertisements are on the header, etc. Thus, the importance of blocks can be assigned according to their relative location in the page. This can be achieved, for instance, method [57]. Based on extracted spatial and content features of blocks, they use supervised machine learning algorithms to assign automatically importance value for each block. We can, also, take into account other parameters to evaluate the importance of a block with respect to the history of changes on this block. For instance, we can consider that the more frequent a block is changing, the less important it is. We are currently looking for the best technique to estimate the importance of blocks.
- *Operations Importance*: The importance of operations depends on the operation type (move, insert, etc.) and the changed element (link, image, etc.). For instance, insert or delete operations can be considered more important than a move. Also, inserting an image can be more important than inserting a link or a text. Again, we plan to study machine learning methods to choose the best parameters values for each operation type.
- *Changes Amount per Block*: The amount of change operations (delete, insert, etc.) occurred inside a block for each element (link, image and text) is deduced from the generated Vi-Delta. This amount represents the percentage of change operations detected for each block divided by the total number of block's elements.

Based on these parameters, the following function  $E(v1, v2)$  to estimates the importance of changes between versions  $v1$  and  $v2$ , each composed of blocks  $Bk_i$ :

$$E = \sum_{i=1}^{NBk} I(Bk_i) * \left[ \frac{1}{NOp} \sum_{j=1}^{NOp} I(OP_j) * \frac{1}{NEl} \sum_{k=1}^{NEl} \frac{N(OP_j, El_k)}{N(El_k, bk_i)} \right]$$



Where:

-  $Op_j = \{\text{insert, delete, update, move}\}$

-  $El_k = \{\text{link, image, text}\}$

-  $N_{El}$ ,  $N_{Op}$ ,  $N_{Bk}$  are respectively the number of elements type, operation type and blocks in the page.

-  $I(x)$  denotes the importance value of  $x$  which can be a block or a change operation. In order to normalize the result of function  $E()$ , we add the following constraint on the importance of blocks

$$: \sum_{i=1}^{N_{Bk}} I(Bk_i) = 1; 0 \leq I(Op) \leq 1$$

-  $N(Op_j, El_k)$  denotes the number of change operation  $j$  that occurred on the element  $k$ .

-  $N(El_k, Bk_i)$  denotes the total number of elements  $k$  inside the block  $i$ .

The function  $E()$  is computed by multiplying the percentage of changes, for each operation ( $Op_j$ ) and block  $Bk_i$ , by the importance of operations  $I(Op_j)$  and blocks  $I(Bk_i)$ . It returns a normalized value between 0 and 1.

Let's take an example, given the blocks importance as shown in Figure 57, the change operations, detected in a delta between two versions of pages, are:

- (i) An update of a text in block B1
- (ii) An insertion of 4 links in block B2.2
- (iii) A deletion of 2 images in block B3.

In this example, the operations insert and update are considered more important ( $I(\text{insertion}) = I(\text{update}) = 1$ ) than a delete ( $I(\text{deletion}) = 0.8$ ). In the old version of the page, the block B1 has one text element, B2.2 has 2 links and B3 has 4 images.

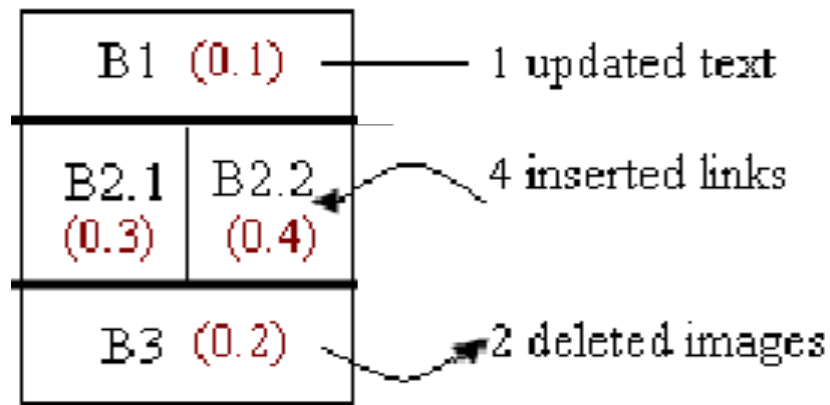


Figure 57 Changes importance example

The importance of changes is computed as follows:

$$\begin{aligned}
 E &= I(Bk1) \cdot I(\text{update}) \cdot [N(\text{update}, \text{text}) / N(\text{text}, Bk1)] + I(Bk2.2) \cdot I(\text{insertion}) \cdot [N(\text{insertion}, \text{link}) / N(\text{link}, Bk2)] + I(Bk3) \cdot I(\text{deletion}) \cdot [N(\text{deletion}, \text{image}) / N(\text{image}, Bk3)] \\
 &= 0.1 * 1 * (1/1) + 0.4 * 1 * (4 / (2+4)) + 0.2 * 0.8 * (2 / 4) = 0.44
 \end{aligned}$$

## CONCLUSION

Unified V-model for re-engineering provides effective and easy way of reconstruction and re-testing web application, it also provides flexibility and reusability that takes into consideration all the user and business standard. The proposed V-model of re-engineering process is proposed for designing the testing protocol for web application development. By introducing V- model into web reengineering process, it increases the maintainability and the effectiveness of the website because V-model led to better validation and verification and produce in accordance to web development life cycle. The structure of V- model enables testing process starts from unit testing to acceptance testing. Re-engineering V-model will save the time for reconstructing or refactoring the web due to strong testing and validation

Proposed a STAR paradigm: Situation, Tools, Application and Restructuring that support the comprehension of existing web application during maintenance. The aim of the paradigm is to reconstruct web application illustrating various aspects of web reengineering. It provides effective and easier way for the reconstruction of web application. Introduction of STAR paradigm into web reengineering process can increase maintainability and effectiveness as it led to better understanding of procedures and practices. STAR paradigm will save the time for re-constructing or restructuring the web application due to strong discerning and contrivance.

In this report a new approach for extracting web content structure based on visual representation was proposed named as Dynamic Data Extraction Algorithm. The produced web content structure is very helpful for applications such as web adaptation, information retrieval and information extraction. By identifying the logic relationship of web content based on visual layout information, web content structure can effectively represent the semantic structure of the web page. An automatic top-down, tag-tree independent and scalable algorithm to detect web content structure was presented. It simulates how a user understands the layout structure of a web page based on its visual representation. DDE Algorithm also removes noise from the extracted web content structure. Apply Cost Constructive Model in the Web Application to estimate the maintenance cost of the web page, calculate the Web size metrics such as lines of code of Web content structure, and then calculate development time and maintenance cost. Comparison of the maintenance cost with and without noisy dynamic web page has been done in all three modes of Cost Constructive Model with the help of DDE algorithm. We conclude that 20 percent maintenance cost will inclined in dynamic Web page with noise as compared to without noise for Organic mode (small number of lines of code), around 50 percent

maintenance cost will inclined for Semi-detached mode and around 80 percent maintenance cost will inclined for Embedded mode. If Number of NoT increases, size of noisy data increases that will directly affect the maintenance cost. Lastly we optimizing the consistency using DDE algorithm.

## References

- [1] P. Fraternali, Tools and Approaches for Developing Data-Intensive web Applications: a Survey, *ACM Computing Surveys*, 1999.
- [2] S. Selmi, N. Kraiem, and H. Ben Ghezala, toward a comprehension view of web engineering, 2005.
- [3] E. Chikofsky and J.H.Cross, Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software Engineering journal*, (Jan. 1990), pp 13-17.
- [4] IEEE Std 1219-1998, In IEEE Standards Software Engineering, 1999 Edition, Volume Two, Process Standards, IEEE Press.
- [5] [Jovanovic, N.](#) ; Secure Syst. Lab., Tech. Univ. of Vienna ; [Kruegel, C.](#) ; [Kirda, E.](#), a static analysis tool for detecting Web application vulnerabilities, Security and Privacy, 2006 IEEE.
- [6] [X. Zhang](#) ; Comput. & Inf. Eng. Coll., Hohai Univ., Nanjing, China ; [Z. Wang](#), e-Business and Information System Security (EBISS), 2010 2nd International Conference on 22-23 may 2010.
- [7] J. Conallen, Building Web Applications with UML, Addison- Wesley Publishing Company, Reading, MA, 1999.
- [8] J. Conallen, Modelling web application architectures with uml, *Communications of the Association for Computing Machinery*, 42(10), October 1999.
- [9] J. Conallen, Modelling web application with uml, White paper, Conallen Inc.  
<http://www.conallen.com/whitepapers/webapps/Modellingwebapplication.htm>, March 1999.
- [10] I. Jacobson, G. Booch and J. Rumbaugh (1999), “The Unified Software Development Process”. Addison Wesley Longman, Inc, Reading: MA.
- [11]. Hassan AE, Holt RC. Towards a better understanding of Web applications. *Proceedings 3rd International Workshop on Web Site Evolution*. IEEE Computer Society Press: Los Alamitos CA, 2001; 112–116.
- [12]. Martin J, Martin L. Web site maintenance with software engineering tools. *Proceedings 3rd International Workshop on Web Site Evolution*. IEEE Computer Society Press: Los Alamitos CA, 2001; 126–131.

- [13]. M'uller HA, Klashinsky K. Rigi—A system for programming in the large. *Proceedings International Conference on Software Engineering*. IEEE Computer Society Press: Los Alamitos CA, 1988; 80–86.
- [14]. Chung S, Lee YS. Reverse software engineering with UML for Web site maintenance. *Proceedings 1st International Conference on Web Information Systems Engineering*. IEEE Computer Society Press: Los Alamitos CA, 2001; 157–161.
- [15]. Ricca F, Tonella P. Web site analysis: Structure and evolution. *Proceedings International Conference on Software Maintenance*. IEEE Computer Society Press: Los Alamitos CA, 2000; 76–86.
- [16]. Ricca F, Tonella P. Understanding and restructuring Web sites with ReWeb. *IEEE Multimedia* 2001; **8**(2):40–51.
- [17]. Schwabe D, Esmeraldo L, Rossi G, Lyardet F. Engineering Web applications for reuse. *IEEE Multimedia* 2001; **8**(1):20–31.
- [18]. Boldyreff C, Kewish R. Reverse engineering to achieve maintainable WWWsites. *Proceedings Eighth Working Conference on Reverse Engineering*. IEEE Computer Society Press: Los Alamitos CA, 2001; 249–257.
- [19]. Vanderdonckt J, Bouillon L, Souchon N. Flexible reverse engineering of Web pages with VAQUISTA. *Proceedings 8<sup>th</sup> Working Conference on Reverse Engineering*. IEEE Computer Society Press: Los Alamitos CA, 2001; 241–248.
- [20]. Di Lucca GA, Di Penta M, Antoniol G, Casazza G. An approach for reverse engineering of Web-based applications. *Proceedings 8th Working Conference on Reverse Engineering*. IEEE Computer Society Press: Los Alamitos CA, 2001; 231–240.
- [21]. Di Lucca GA, Fasolino AR, De Carlini U, Pace F, Tramontana P. WARE: A tool for the reverse engineering of Web Applications. *Proceedings 6th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society Press: Los Alamitos CA, 2002; 241–250.
- [22]. Di Lucca GA, Fasolino AR, Faralli F, De Carlini U. Testing Web applications. *Proceedings International Conference on Software Maintenance*. IEEE Computer Society Press: Los Alamitos CA, 2002; 310–319.

- [23]. Kirchner M. Evaluation, repair, and transformation of Web pages for Web content accessibility. Review of some available tools. *Proceedings 4th International Workshop on Web Site Evolution*. IEEE Computer Society Press: Los Alamitos CA, 2002; 65–72.
- [24]. Tonella P, Ricca F, Pianta E, Girardi C. Restructuring multilingual Web sites. *Proceedings International Conference on Software Maintenance*. IEEE Computer Society Press: Los Alamitos CA, 2002; 290–299.
- [25]. Paganelli L, Paterno F. Automatic reconstruction of the underlying interaction design of Web applications. *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE02)*. ACM Press: New York, 2002; 439–445.
- [26]. Shefali Singhal and Neha Garg. Article: Hybrid Web-page Segmentation and Block Extraction for Small Screen Terminals. IJCA Proceedings on 4th International IT Summit Confluence 2013.
- [27]. Anjali singh, Web Content Extraction to Facilitate Web Mining. International Journal of Electronics and Computer Science Engineering, ISSN- 2277-1956.
- [28]. C.V.S.R SYAVASYA, An Approach to Find Maintenance Costs Using Cost Drivers of Cocomo Intermediate Model. International Journal of Computational Engineering Research (ijceronline.com) Vol. 3Issue. 1.
- [29] B. Laurent, L. Quentin, V. Jean and M. Benjamin, “Reverse Engineering of Web Pages Based on Derivation and Transformation” , 27 August 2005.
- [30] G. K.Tyagi and D. P. Ballou, “Examining Data Quality”, communication ACM, 41 (2), Feb. 1998.
- [31] Stanjarzabek, “Strategic reengineering of software:life cycle approach”, Department of Information Systems and Computer Science National University of Singapore.
- [32] S. Jarzabek, “Domain Model-Driven Software Reengineering and Maintenance”, Journal of Systems andSoftware, January 1993, pp. 37-51.
- [33] M. Giordano, G. Polese, G. Scanniello, and G. Tortora, “Visual Modelling of Role-Based Security Policies in Distributed Multimedia Applications”. In Proc. of IEEE6th International Symposium on Multimedia SoftwareEngineering, Miami, FL, USA, IEEE CS Press, 2004, pp.:138 – 141.

- [34] D. Distanto, T. Parveen, and S. Tilley, "Towards a Technique for Reverse Engineering Web Transactions from a User's Perspective", In Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC 2004: June 24-26, 2004; Bari, Italy). Los Alamitos, CA: IEEE CS Press, 2004.
- [35] S. Tilley, D. Distanto, and S. Huang, "Design Recovery of Web Application Transactions", Submitted to the 11th IEEE Working Conference on Reverse Engineering (WCRE 2004: Nov. 9-12, Delft, the Netherlands). June 2004.
- [36] P. Szekely, P. Luo, and R. Neches, "Beyond Interface Builders: Model-Based Interface Tools", Proc. of ACM Conf. on Human Aspects in Computing Systems InterCHI'93, ACM Press, New York, 1993, pp. 383-390.
- [37] J. Vanderdonckt and P. Berquin, "Towards a Very Large Model-based Approach for User Interface Development", Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99, IEEE Computer Society Press, Los Alamitos, 1999, pp. 76-85.
- [38] Offutt J, "Quality attributes of Web software applications", IEEE Software 2002; 19(2):25-32.
- [39] P. Diman, N. Singh and K. Kumar, "Unified V- Model Approach of Reengineering to reinforce Web Application Development", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727 Volume X, Issue X (Nov. - Dec. 2013), PP 01-00.
- [40] M. Moore, "Representation Issues for Reengineering Interactive Systems", ACM Computing Surveys Special issue: position statements on strategic directions in computing research, Vol. 28, No. 4, Dec 1996, article # 199, ACM Press, New York, NY, USA.
- [41] M. Moore and S. Rugaber, "Using Knowledge Representation to Understand Interactive Systems", in Proc. of the Fifth International Workshop on Program Comprehension IWPC'97 (Dearborn, 28-30 May 1997), IEEE Computer Society Press, Los Alamitos, 1997.
- [42] G. Mori, F. Paternò, C. Santoro, "Tool support for designing nomadic applications", Proc. of the 2003 international conference on Intelligent user interfaces, Jan 2003, (Miami, USA), ACM Press, New York, USA, pp141-148.



- [43] Di Lucca, G.A., Fasolino, A.R., Pace F., Tramontana, P. and De Carlini, U. (2002a), "WARE: A Tool for The Reverse Engineering of Web Applications", Proc. 6th European Conference on Software Maintenance and Reengineering (CSMR'02), pp241-250.
- [44] D. Lucca, G.A., Fasolino, A.R. and Tramontana, P. (2004), "Reverse Engineering Web Applications: The WARE Approach", Journal of Software Maintenance and Evolution, Research and Practice, Vol 16, pp71-101.
- [45] D. Draheim, E. Fehr and G. Weber (2003), "JSPick – A Server Pages Design Recovery", Proc 7th IEEE European Conference on Software Maintenance and Reengineering, LNCS, pp230-236.
- [46] Jovanovic, N. ; Secure Syst. Lab., Tech. Univ. of Vienna ; Kruegel, C. ; Kirda, E., "a static analysis tool for detecting Web application vulnerabilities", Security and Privacy, 2006 IEEE.
- [47] E. Chikofsky and J.H. Cross, "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software Engineering journal, (Jan. 1990), pp 13-17.
- [48] IEEE Std 1219-1998, In IEEE Standards Software Engineering, 1999 Edition, Volume Two, Process Standards, IEEE Press.
- [49] Di Lucca, G.A., Fasolino, A.R. ; Pace, F. ; Tramontana, P. ; De Carlini, U. Comprehending Web applications by a clustering based approach, Program Comprehension, 2002. Proceedings. 10th International Workshop on Clustering approach, IEEE Computer Society Press: Los Alamitos CA, 2001; ISSN: 1092-8138; 261 – 270.
- [50] M. Moore, "Representation Issues for Reengineering Interactive Systems", ACM Computing Surveys Special issue: position statements on strategic directions in computing research, Vol. 28, No. 4, Dec 1996, article # 199, ACM Press, New York, NY, USA.
- [51] Chen, Jinlin and Zhou, Baoyao and Qiu, Fengwu. Function- based object model towards website adaptation. In 10th Int'l World Wide Web Conference, May, 2001.
- [52] Kovacevic, Milos and Diligenti, Michelangelo and Gori, Marco, Milutinovic, Veljko. Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification. IEEE International Conference on Data Mining (ICDM' 02), December, 2002

- [53]. Cai, Deng. VIPS: a Vision based Page Segmentation Algorithm. Microsoft Technical Report (MSR-TR-2003-79), 2003
- [54]. Tang, Y. Y., Cheriet, M., Liu, J., Said, J. N., and Suen, C. Y., Document Analysis and Recognition by Computers, Handbook of Pattern Recognition and Computer Vision, edited by C.
- [55]. Yi, L., Liu, B., and Li, X. Eliminating Noisy Information in Web Pages for Data Mining. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, DC, pp. 296–305, 2003.
- [56]. Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. Extracting Content Structure for Web Pages Based on Visual Representation. In Web Technologies and Applications. Springer, pp. 406–417, 2003.
- [57]. R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In WWW '04: Proceedings of the 13th international conference on World Wide Web, 2004.