RESTRUCTURING OF WEB APPLICATIONS BASED ON PARTICLE SWARM OPTIMIZATION USING WEB PAGE RELATIONSHIPS AND METRICS

A dissertation report submitted in the partial fulfillment for the award of degree of

Master of Technology

in

Software Engineering

By

Chaman Singh Charnawat

Roll No. 04/SWE/2K10

Under the esteemed guidance of

Ms. Abhilasha Sharma

Department of Computer Engineering

Delhi Technological University, Delhi-42

2011-12

CERTIFICATE

Delhi Technological University

(Govt. Of National Capital Territory of Delhi)

Bawana Road, Delhi – 110042

Date: ___________________

This is to certify that the thesis entitled '**Restructuring Of Web Applications Based On Particle Swarm Optimization Using Web Page Relationships And Metrics**' done by **Chaman Singh Charnawat (04/SWE/2K10),** for the partial fulfillment of the requirements for the award of the degree of Masters of Technology in Software Engineering, is an authentic work carried out by him under my guidance.

Project Guide:

Ms. Abhilasha Sharma

Assistant Professor, Department of Software Engineering

Delhi Technological University, Delhi 110042

ACKNOWLEDGEMENT

I would like to take this opportunity to express my profound sense of gratitude and respect to all those who have helped me throughout the duration of my work.

I would like to thank Ms. Abhilasha Sharma, Assistant Professor, Department of Software Engineering, DTU, Delhi for her benevolent guidance in completing my project work titled "Restructuring of Web Applications Based on Particle Swarm Optimization Using Web Page Relationships and Metrics". Her kindness and help have been the source of encouragement for me, without which this work would have been a dream for me.

Also I would like to extend my thanks to the entire faculty of the Department of Computer Engineering, DTU, Delhi for their valuable guidance wherever and whenever required.

I must not forget to give sincere regards to my revered parents for their constant support, encouragement, understanding and love without which it would have been impossible for me to achieve all that I have.

Last but not the least I like to thank all the concerned ones who directly or indirectly helped me during the tenure of my work.

Chaman Singh Charnawat

Master of Technology (Software Engineering)

Roll No. 04/SWE/2010

Delhi Technological University, Delhi 110042

ABSTRACT

With the number of web applications on the internet increasing rapidly it has become very important to engineer them in ways similar to how we develop and maintain mainstream software. Most of the existing web applications were developed using ad hoc techniques and as a result, most of them are suffering from various structural flaws. This increases the effort and cost required to maintain the application and decreases component reuse.

In this study titled '*Restructuring Of Web Applications Based On Particle Swarm Optimization Using Web Page Relationships And Metrics*' we tried to improve the quality of existing web applications by removing structural flaws that exist in its implementation.

Previous research has used genetic algorithm (GA) to optimize a web application and used coupling and cohesion to create clusters of web pages. The results showed that changing the structure provide an impact on the quality of the application. We extend that work by using particle swarm optimization for restructuring purposes in an attempt to improve the quality of the web application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

In today's world, which is highly dependent on web based services, web applications are increasing rapidly on the Internet and the frequency at which they see change has also increased due to,

- Changes to the stored information
- Maintenance tasks
- Addition and deletion of functions, etc

Many web applications are still implemented using static HTML pages in which data and layout information are interleaved. This leads to out-of-date information, inconsistent style, tricky and expensive maintenance and the structure of the applications is also not optimal. Due to these structural flaws cost, time, and effort required to maintain the application increases and component reuse decreases.

Given the complexity, modern web applications can be seen as software systems running on the web comprising thousands of line of code split into many modules. Due to the pressing market demand, new web applications are usually developed in very short time, while existing applications are modified frequently and quickly. In such situations, the existing software engineering principles are not usually applied, as well as existing software processes and methodologies are rarely adopted. As a consequence, web applications present disordered architectures, poor or non-existing documentation.

Web restructuring methods and tools are being proposed in order to reduce the effort required to comprehend existing web applications and to support their maintenance and evolution.

1.2 Motivation

By using restructuring or clustering of web pages or web sites, effort and time required for maintenance can be reduced. There are various web pages in a single web application and somehow they are related to each other by the content, link, usages etc. If these web pages can be categorized on the basis of their similarities then the structural flaws of web application could decrease to some extent. Researchers have previously used genetic algorithm (GA) to restructure a web application and used coupling and cohesion to create clusters of web pages. The results showed that changing the structure has an impact on the quality of the application.

1.3 Statement of Work

In this study we try to improve the quality of existing web applications by removing structural flaws that exist in its implementation. We extend the previously done work by using particle swarm optimization (PSO) for restructuring purposes in an attempt to improve the quality of the web application.

1.4 Organization of Report

The report is organized in the following manner,

- Chapter 2 describes in detail all the work done in the area of web restructuring, in particular all the work done and terminology related to our research.

- Chapter 3 describes overview of software metrics, web metrics and used metrics in our work.
- Chapter 4 highlights the technique proposed with a detailed description of the PSO algorithm and how it is better than previously applied techniques like genetic algorithm(GA). Result are listed and analyzed at the end of this chapter.
- Conclusion and future work in the research area along with the limit are mentioned further followed by the list of the references.

Chapter 2

LITERATURE SURVEY

2.1 What is Web Engineering?

Web engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality web-based systems and applications [1].

As an emerging discipline, web engineering actively promotes systematic, disciplined and quantifiable approaches towards successful development of high-quality, ubiquitously usable web-based systems and applications [1].

In the absence of a disciplined approach to web based system or web application development, we found that web based applications are not delivering desired performance and quality, and that development process also becomes increasingly complex and difficult to manage and refine and also expensive and grossly behind schedule. Web engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Internet- and web based systems.

Nowadays the growth of the Internet, Intranets, Extranets, and the World Wide Web has already had a significant impact on every sector like business, commerce, industry, banking and finance, education, government and entertainment sectors, and our personal and working life. Many legacy information and database systems are being migrated to the Internet in the form of web applications. Large of new, complex distributed applications is emerging in the web environment. The popularity and ubiquity stems from the nature of the web itself and its features:

web provides an information representation that supports interlinking of all kinds of content, easy access for end users, and easy content creation using widely available tools. These are the benefits of the web or web applications.

Web engineering is multidisciplinary and encompasses contributions from diverse areas such like systems analysis and design, software engineering, hypermedia/hypertext engineering, requirements engineering, human-computer interaction, user interface, information engineering, information indexing and retrieval, testing, modeling and simulation, project management, and graphic design and presentation. Web engineering and software engineering both involve programming and software development.

While web engineering uses software engineering principles, it encompasses new approaches, methodologies, tools, techniques, and guidelines to meet the unique requirements of web based applications [2].

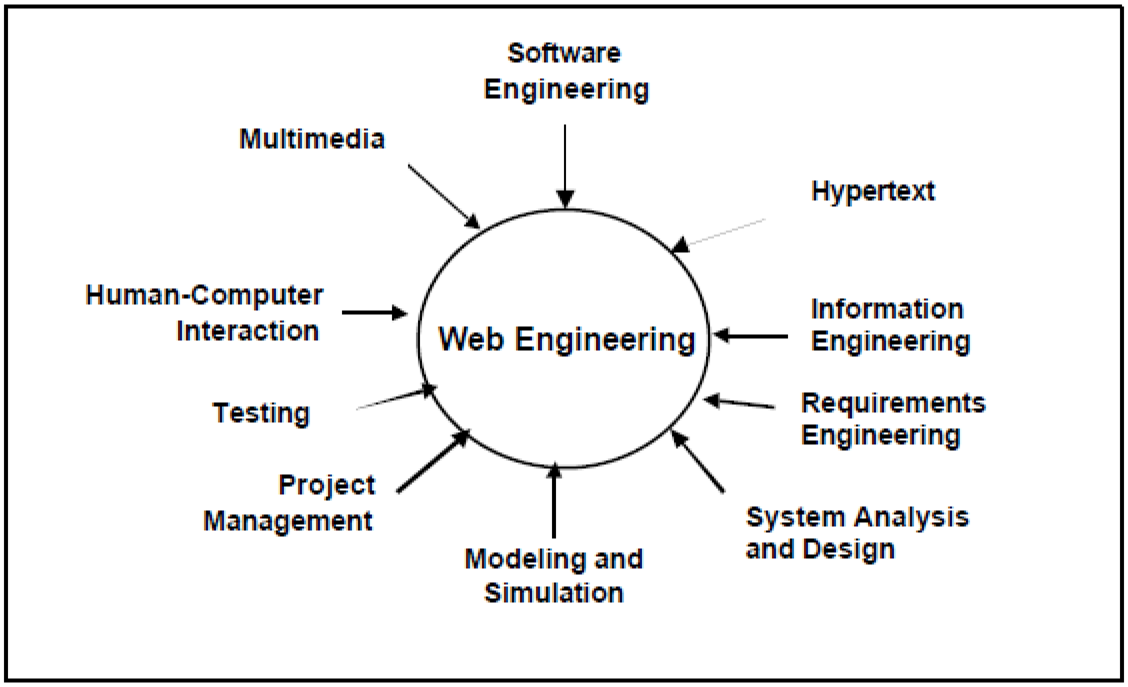

Figure 2.1: Web Engineering – a multidisciplinary field[2]

2.1.1 Web Engineering Activities

Following are the major web engineering activities.

- Requirements specification and analysis
- Web based system development methodologies and techniques
- Integration with legacy systems
- Migration of legacy system to web environments
- Web based real-time applications development
- Testing, verification and validation
- Quality assessment, control and assurance
- Configuration and project management
- Web metrics for estimation of development efforts
- Performance specification and evaluation
- Update and maintenance
- Development models, teams, staffing
- Human and cultural aspects
- User centric development, user modeling and user involvement and feedback
- End user application development
- Education and training

2.1.2 What are web applications?

A web application is any application that uses a web browser as a client. The application can be as simple as a message board or a guest sign in book on a website, or as complex as a word processor or a spreadsheet [2].

A web application is an application that is accessed over a network such as the Internet or an intranet [3].

Web applications are popular because of the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The key reason for their popularity is the ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers, as is the inherent support for cross-platform compatibility. Generally common web applications include webmail, online retail sales, online auctions, wikis and many other functions. Examples of browser applications are simple office software (word processors, and presentation tools, online spreadsheets), but can also include more advanced applications such as project management, computer-aided design, video editing.

There are various benefits of the web application and some of them are following:

- Web applications do not require any complex "roll out" procedure to deploy in large organizations. A compatible web browser is all that is needed.
- Browser applications typically require little or no disk space on the client.
- They require no upgrade procedure since all new features are implemented on the server and automatically delivered to the users.
- Web applications integrate easily into other server-side web procedures, such as email and searching.
- They also provide cross-platform compatibility in most cases (i.e., Windows, Mac, Linux, etc.) because they operate within a web browser window.

With the advent of HTML5, programmers can create richly interactive environments natively within browsers which also included the list of new features like native audio, video and animations, as well as improved error handling.

2.1.3 What is web restructuring and web reengineering?

Restructuring is the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behavior (functionality and semantics) [4].

Reengineering, also known as both renovation and reclamation, is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form [4].

A restructuring transformation is often one of appearance, such as altering code to improve its structure in the traditional sense of structured design. The term "restructuring" came into popular use from the code-to code transform that recasts a program from an unstructured form to a structured form. However, the term itself has a broader meaning that recognizes the application of similar transformations and recasting techniques in reshaping data models, design plans, and requirements structures. Data normalization, for example, is a data to data restructuring transform to improve a logical data model in the database design process.

Some types of restructuring can be performed with knowledge of structural form but without an understanding of meaning. For example, you can convert a set of if, else statements into a case structure, or vice versa, without knowing the program's purpose or anything about its problem domain.

While restructuring creates new versions that implement or propose change to the subject system, it does not normally involve modifications because of new requirements, However, it may lead to better observations of the subject system that suggest changes that would improve aspects of the system. Restructuring is often used as a form of preventive maintenance to improve the physical state of the subject system with respect to some preferred standard. It may also involve adjusting the subject system to meet new environmental constraints that do not involve reassessment at higher abstraction levels.

Reengineering, also known as both renovation and reclamation, is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. reengineering generally includes some form-of reverse engineering (to achieve a more abstract description) followed by some form of forward engineering or restructuring. This may include modifications with respect to new requirements not met by the original system. For example, during the reengineering of information- management systems, an organization generally reassesses how the system implements high level business rules and makes modifications to conform to changes in the business for the future. There is some confusion of terms, particularly between reengineering and restructuring. The IBM user group Guide, for example, defines “application reengineering” as “the process of modifying the internal mechanisms of a system or program or the data structures of a system without changing the functionality *(sys*tem capabilities as perceived by the user)”. In other words, it is altering the how without affecting the *what.* This is closest to our definition of restructuring. However, two paragraphs later, the same publication says, “It is rare that an application is reengineered without additional functionality being added.” This supports our more general definition of reengineering. While reengineering involves both forward engineering and reverse engineering, it is not a super type of the two. Reengineering uses the forward and reverse engineering technologies available, but to date it has not been the principal driver of their progress. Both technologies are evolving rapidly, independent of their application within reengineering.

2.1.4 What is the Software Metrics and Web Metric?

IEEE Standard 1061 [1] lays out a methodology for developing metrics for software quality attributes. The standard defines an *attribute* as "a measurable physical or abstract property of an entity." A *quality factor* is a type of attribute, "a management oriented attribute of software that contributes to its quality." A metric is a measurement function, and a software quality metric is "a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality." The IEEE Standard 1061 answer lies in the use of direct metrics. A *direct metric* is a metric that does not depend upon a measure of any other attribute.

The contrast between direct measurement and indirect, or derived measurement, is between a (direct) metric function whose domain is only one variable and a (derived) function whose domain is an n-tuple. For example, density is a function of mass and volume. Some common derived metrics in software engineering are programmer productivity (code size/ programming time), module defect density (bugs / module size), requirements stability (number of initial requirements / total number of requirements), system spoilage (effort spent fixing faults / total project effort).

Software metric is a measure of some property of a piece of software or its specifications. Software metrics is the rather misleading collective term used to describe the wide range of activities concerned with measurement in software engineering [6]. Metrics are standards that define measurable attributes of entities, their units and their scopes. Metrics, especially those measuring phenomena, are invariably proposed in the context of techniques for improving the quality and usefulness of measurable objects [7].

One of the earliest attempts to make global measurements about the web was undertaken by Bray [1996]. A key element of any web site engineering process is metrics. Web metrics are used to better understand the attributes of the web page we create. But, most important, we use web metrics to assess the quality of the web engineered product or the process to build it. Since metrics are crucial source of information for decision making, a large number of web metrics have been proposed in the last decade to compare the structural quality of a web page [7].

Web metrics can be categorized as followed.

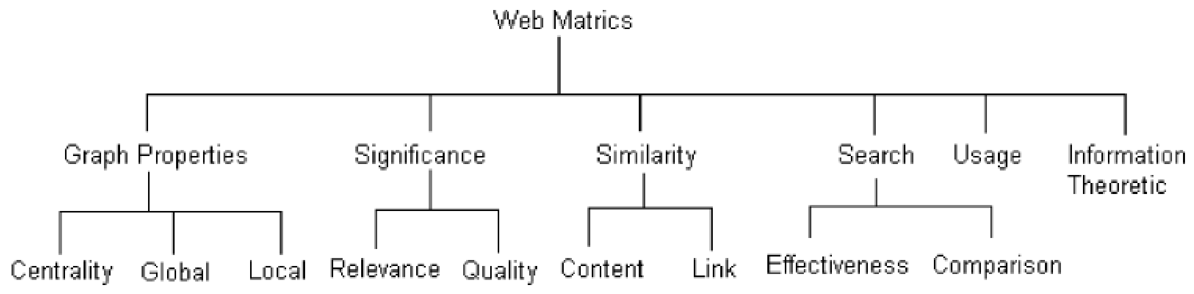

Figure 2.2: A taxonomy of web metrics[7]

2.1.4.1 Similarity Web Metrics:

Web page similarity metrics measure the extent of relatedness between two or more web pages. Similarity functions have mainly been described in the context of web page clustering schemes.

Clustering is a natural way of semantically organizing information and abstracting important attributes of a collection of entities. Clustering has certain obvious advantages in improving information quality on the WWW. Clusters of web pages can provide more complete information on a topic than individual pages, especially in an exploratory environment where users are not aware of several pages of interest. Clusters partition the information space such that it becomes possible to treat them as singular units without regarding the details of their contents.
similarity metrics classified into i) content based, ii) link based and iii) usage based metrics.

i) Content-Based Similarity Metrics: Content-based similarity is measured by comparing the text of documents. Pages with similar content may be considered topically related and designated the same cluster.

ii) Link-Based Similarity Metrics: Link-based measures rely exclusively on the hyperlink structure of a web graph to obtain related pages.

iii) Usage-Based Similarity Metrics: Usage based similarity is based on patterns of document access. The intent is to group pages or even users into meaningful clusters that can aid in better organization and accessibility of web sites.
For grouping interacting various related pages, here, define link-based page similarity (PS), based on the cosine equation, which is derived from page relationship weights.

$$PS(u,v) = \frac{\sum_{w \in WP} W(u,w) * W(v,w)}{\sqrt{\sum_{w \in WP} W(u,w)^2 * \sum_{w \in WP} W(v,w)^2}} \ .$$

where *WP* is a set of web pages.

We can also define cohesion and coupling metrics using page similarity based on connection strength considering the type and the number of the relationships and the number of parameters transferred between web pages.

$$COH(C) = \frac{\sum_{u,v \in C} PS(u,v)}{(|C| * |C|)} \text{ .}$$

[1]

where |C| is the number of pages in cluster C.

Cohesion is a measure of how strongly-related each piece of functionality expressed by the source code of a software module is or cohesion is a measure of how strongly-related or focused the responsibilities of a single module are.

$$COP(C_i, C_j) = \sum_{u \in C_i} \sum_{v \in C_j} PS(u,v) \text{ .}$$

[1]

Where |C| is the number of pages in cluster C.

In software engineering, *coupling or dependency* is the degree to which each program module relies on each one of the other modules. Coupling can be "low" (also "loose “and "weak") or "high" (also "tight" and "strong").

Coupling talks about the interdependencies between the various modules while cohesion describes how related functions within a module are. Low cohesion implies that module performs tasks which are not very related to each other and hence can create problems as the module becomes large.

2.2 Study Carried Out In Literature

Various research work carried out in the field of web restructuring, different researchers have used different techniques for web restructuring.

2.2.1 Byungjeong Lee, Eunjoo Lee, and **ChisuWu proposed** a web optimization technique based on genetic algorithm where they used similarity metrics to calculate the page similarity so that cohesion and coupling could be calculating. After calculating cohesion and coupling, they calculated the clustering objective, and then they applied a genetic algorithm based clustering approach which uses agglomerative clustering with backtracking where each solution is represented by a chromosome (web page).

2.2.2 Filippo Ricca, Paolo Tonella , Ira D. Baxter proposed a technique for restructuring web applications via transformation rules in this technique their idea apply rewrite rules for web applications with the aim of restructuring them because during the evolution phase, the structure (pages and links) of a web application tends unavoidably to degrade. A solution to reverse this degradation could be restructuring the web application, but this work may take a lot of time and effort if conducted without appropriate tools. In this technique a set of automatic and semi-automatic transformation rules are used for improving the quality of the web applications. These rules work at two levels: i) inter page transformations, ii) intra-page transformations.

In the inter page transformations , rules take more pages the entire web application (or a portion of it) – in input, change some links/pages, add/delete some pages, and, at the end of the transformation, return the changed pages.

In the intra-page transformations, rules transform a single HTML page and the result is a new HTML page. There is no impact on the web application model.

The transformations proposed for web applications were targeting to improving 3 quality factors.

i) Maintainability ii) Usability iii) Portability

They have identified six classes of opportunities for rewriting HTML documents and restructuring web applications: i) Syntactic clean up, ii) Design restructuring, iii) Page restructuring, iv) Style renovation and grouping, v) Improving accessibility, vi) Update to new standards.

The first three classes are associated with maintainability, the fourth and fifth ones with improving usability, and the last one is related to portability. Among the identified classes, only design restructuring includes a set of inter page transformations, being almost all the others related to a single page (intra-page transformations).

Syntactic clean up, that corrects the HTML code so that it matches, where possible, the rendering observed in popular browsers.

The rules for design restructuring are related to restructuring a portion of the structure of a web application. Examples are the automatic *reorganization into frames*, *page extraction* and *page insertion*. The purpose of the page restructuring rules is improving the quality of a single HTML page, which is already syntactically correct (compare with syntactic clean up). Examples are: transforming absolute URLs into relative URLs by means of the tag BASE (after this transformation it is possible moving the page between directories and even servers without problems).

Style renovation and grouping rules transform (if possible)an HTML page with presentation markups into a page with style sheets, and organize style sheets in a compact way.

The improving accessibility class contains a set of rules that implement some guidelines, for example, removing the blinking effect, providing the ability to stop the refresh, providing redundant text links for each active region of a server-side image map and providing keyboard shortcuts to important links. Updating to new standards transformation rules substitute the deprecated tags with tags of the latest version of HTML.

By applying these transformation rules they improved the three quality factors maintainability, usability, portability i.e in this paper they have presented some initial ideas on transformation rules working on web applications with the aim of supporting reengineering.

2.2.3 Ching-Yi Cheo and Fun Ye in this paper they describe the methodology for clustering using *particle* swarm optimization. [8] proposed a algorithm which is as follows ,

Stepl) Initialize positions vector X and associated velocity *V* of all particles in the population randomly. Here the position

X of the particle is the center position of each cluster.

Step2) Evaluate the fitness function for each particle. The method is assign point ***x i ,*** i = 1,2 ,..., N to cluster *Cj* , *j *(1*,2, ..., *K)* iff

$$\|\mathbf{x}_i - z_j\| * \| x_i - z_p\| \; p=1,2,\ldots,K, \text{ and } J*p. \qquad (1)$$

And the fitness function for the PSO-clustering is given by :

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} \|x_i - z_j\|^2 \qquad (2)$$

$$fitness = k / (J + J_0) \qquad (3)$$

where the k is a positive constant, and $J,$ is a small-valued constant.

Step3) Compare particle's fitness evaluation with particle's best solution $P,$. If current value is better than $P;$, then set $P,$ value equal to the current value, and the P, position equal to the current position in n-dimensional space.

Step4) Compare fitness evaluation with the population's overall previous best. If current value is better than the $P,$ (the global version of the best value), then reset P, to the current particle's value and position.

Step5) Change velocities and position using equation

$$V_{id}=V_{id} + c_1 * \text{rand}( )*(p_{id}- X_{id}) + c * \text{rand}( ) *(P_{gd} – X_{id})\text{and}$$

$$x_{id} =x_{id}+V_{id..}$$

Step6) Repeat Step2)-Step5) until a stop criterion is satisfied or a predefined number of iterations is completed.

PSO will base on the minimum object function J to search automatically the data cluster centers of n-dimension Euclidean space R'', traditional cluster algorithm such as K-means may get stuck at local optimal solution, depending on the choice of the initial cluster centers. It can't make sure to solve the global optimal solution every time. Related to the other evolution algorithm, PSO needs the less parameter to decide. When it is executed, it can avoid entering the local optimal solution.

2.2.4 Xiaohui Cui, Thomas E. Potok, Paul Palathingal : they use the pso algorithm for document clustering because Fast and high-quality document clustering algorithms play an important role in effectively navigating, summarizing, and organizing information[9]. [9]

presents a Particle Swarm Optimization (PSO) document clustering algorithm. Contrary to the localized searching of the K-means algorithm, the PSO clustering algorithm performs a globalized search in the entire solution space. In the experiments we conducted, we applied the PSO, K-means and hybrid PSO clustering algorithm on four different text document datasets. The number of documents in the datasets ranges from 204 to over 800, and the number of terms ranges from over 5000 to over 7000. The results illustrate that the hybrid PSO algorithm can generate more compact clustering results than the Kmeans algorithm.

In this study, a document clustering algorithm based on the PSO algorithm is proposed. In the PSO clustering algorithm, the clustering behavior can be classified into two stages: the global searching stage and the local refining stage. The global searching stage guarantees each particle searches widely enough to cover the whole problem pace. The refining stage makes all particles converge to the optima when a particle reaches the vicinity of the optimal solution. For a large dataset, conventional PSO can conduct a globalized searching for the optimal clustering, but requires more iteration numbers and computation than the K-means algorithm does. The K-means algorithm tends to converge faster than the PSO algorithm, but usually can be trapped in a local optimal area. The hybrid PSO algorithm combines the ability of globalized searching of the PSO algorithm and the fast convergence of the K-means algorithm and avoids the drawback of both algorithms. The algorithm includes two modules, the PSO module and the K-means module. The PSO module is executed for a short period at the initial stage to discover the vicinity of the optimal solution by a global searching and at the same time to avoid consuming high computation. The result from the PSO module is used as the initial seed of the K-means module. The K-means algorithm is applied for refining and generating the final result. Our experimental results illustrate that using this hybrid PSO algorithm can generate higher compact clustering than using either the PSO or the K-means alone.

2.2.5 G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini : propose an approach based on a clustering method for decomposing a web application (WA) into groups of functionally related components. The approach is based on the definition of a coupling measure between interconnected components of the WA that takes into account both the typology and the topology of the connections. The coupling measure is exploited by a clustering algorithm that produces a hierarchy of clustering. This hierarchy allows a structured approach for the comprehension of the web application to be carried out. The approach has been experimented with medium sized web applications and produced interesting and encouraging results. Because the number and the complexity of web applications are increasing dramatically to satisfy the market requests, and the need of effective approaches for comprehending them is growing accordingly. Recently, some reverse engineering methods and tools have been proposed to support the comprehension of a web application; the information recovered by these tools is usually rendered in graphical representations. However, the graphical representations become progressively less useful with large scale applications, and do not support adequately the comprehension of the application. [10] proposed an approach to support the comprehension of web applications by exploiting clustering techniques . The approach is based on a conceptual model of a WA comprising components and relationships between them, and on a coupling measure between interconnected components that takes into account both the typology and the topology of the connections. The coupling measure is exploited by a hierarchical clustering algorithm for producing a hierarchy of clustering. The hierarchy is used to carry out a structured approach to the comprehension of the web applications. The approach has been experimented with medium sized web applications and produced interesting and encouraging results. In future work, the definition of a finer model of the dependencies between the web application components, that takes into account the data flow too, will be addressed. Moreover, in order to obtain a definition of the coupling between components that takes into account the amount of exchanged data, additional strategies for weighting the dependencies will be defined and experiment with. In order to extend the validity of the experimental results, validation experiments aiming to assess the stability of the clustering approach [11] with the support of suitable metrics [12], or the scalability of the approach, will be carried out.

2.2.6 G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini : [13] presents a tool for reverse engineering web applications. UML diagrams are used to model a set of views that depict several aspects of a web application at different abstraction levels. The recovered diagrams ease the comprehension of the application and support its maintenance and evolution. Because the development of web sites and applications is increasing dramatically to satisfy the market requests. The software industry is facing the new demand under the pressure of a very short time to market and an extremely high competition. As a result, web sites and applications are usually developed without a disciplined process: Web applications are directly coded and no, or poor, documentation is produced to support the subsequent maintenance and evolution activities, thus compromising the quality of the applications.

UML diagrams describing the different views of a web application can be recovered with the support of the reverse engineering tool WARE (Web Application Reverse Engineering) that automatically extracts information from the application and allows more abstract representations to be reconstructed.

WARE is an integrated tool whose architecture comprises the following main components:

(1) Interface Layer

(2) Service Layer

(3) Repository.

The *Interface Layer* implements a user interface providing the access to the functions offered by the tool, and the visualization of recovered information and documentation, both in textual and graphical format.

The *Service Layer* implements the tool services, and includes two main components: the extractor and the abstractor. The former parses the web application source code and produces an intermediate representation form (IRF) that provides a representation of the data items extracted from the web application. The Abstractor operates over the IRF, and implements several abstraction tasks necessary to support the recovery of the UML diagrams of the application. The information produced by the Abstractor is stored in a relational database. The *Repository* includes the IRF, the relational database populated by the abstractor, and the recovered diagrams.

The number and the economic relevance of web applications are incessantly increasing. The maintenance and evolution of web applications will become more and more a dominant task in the future. Unfortunately, web applications usually developed without a disciplined process, and no or poor documentation is produced. This inadequacy makes hard and expensive the maintenance of existing web applications. The lack of documentation forces the maintainers to extract information about the application from its source code and, therefore, reverse engineering techniques and tools are needed to support this task. That's why [13] presented the reverse engineering tool WARE, whose main purpose is to provide a support to the recovery, from existing web applications, of UML diagrams dealing not only with static content, but also with the more challenging dynamic content. The tool has been submitted to validation experiments, carried out for exploring its adequacy to the needs of a maintainer of existing web applications. The experiments produced the following results. The tool provided a precious support for the reconstruction of UML diagrams of undocumented web applications. Some of the required reverse engineering activities were automatically performed by the tool, while other activities were carried out semi-automatically, with the assistance provided by the tool. The UML diagrams obtained could be used to support the subsequent maintenance and evolution of the web applications. We also learned some lessons from these experiments. Some useful indications for improving the effectiveness of the reverse engineering approach supported by the tool emerged. As an example, a criterion for identifying sets of interconnected classes implementing a well defined behavior is to explore the directory structure of the web application, since the files of recovered clusters of related classes often belonged to a same directory. Of course, as shown by other case studies, this criterion cannot be successfully applied when the web application developers did not group the related components into a single directory.

Moreover, since strongly connected components in the recovered class diagram were often involved in the same function, dominance and cluster analysis based criteria could be suggested to identify candidate sets of classes to be associated with a single function. These criteria might be more successful than the directory based one, but of course this hypothesis should be experimentally validated. On the basis of the data collected during the experiments, an effective reverse engineering process supported the WARE tool could be defined. The process includes four main steps:

- web application static analysis and class diagram recovery;
- identification of notable sub-graphs (i.e., sets of classes) in the class diagram, where each sub-graph (set) will be responsible for a web application functionality;
- use cases recovery, by associating each set of classes to a single use case;
- Sequence diagrams recovery, for obtaining several scenarios of using the web application, by analyzing the dynamic interactions among the web application components.

[13]Future work will be addressed to improve the tool in order to reduce its main weakness, that is the limited number of analyzed scripting languages, and the manageability of the diagram visualization.

2.2.7 G.A. Di Lucca, A. R. Fasolino, P. Tramontana: web applications usually present disordered architectures, poor or non-existing documentation, and can be analyzed, comprehended and modified with a considerable effort. Reverse engineering methods and tools are being proposed in order to reduce the effort required to comprehend existing web applications and to support their maintenance and evolution. In this paper, the experimentation of a reverse engineering approach is described. In [14] the experiment carried out will be described, and the major problems encountered in reverse engineering web applications will be presented. A set of recommendations and rules that should be applied for producing more analyzable and maintainable applications could be deduced.

A method for reverse engineering (RE) web applications has been presented in [15, 13]. The method defines a reverse engineering process including distinct activities for obtaining 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 WARE, a reverse engineering tool. The proposed reverse engineering process includes five main steps:

- Static Analysis of the web application
- Dynamic Analysis of the web application
- Automatic Clustering of the web application
- Validation of the clustering
- Abstraction of UML diagrams.

[14] presented an approach for reverse engineering web applications, and illustrated the results of an experiment carried out to assess which characteristics of a web application mostly affect its comprehensibility. During the experiment, the RE process was executed by software engineers with the support of a tool that automates some of the steps of the process. The problems emerged during the experiment were analyzed in order to propose possible strategies for reducing their negative effect on the comprehensibility of the applications. The first lesson we learned from the experiment was that the effort required for comprehending a web application, including script code interleaved with HTML code, may be reduced if the programmers did not abuse the mechanisms offered for obtaining dynamic behavior.

Navigation links should be explicitly distinguished from links implementing functional semantic relationships between different components of the web application, in order to simplify the identification of cohesive groups of pages, implementing a well defined behavior. Moreover, the conceptual structure of the web application should be mirrored by the physical distribution of its components on the file system, and suitable internal documentation standards should be used during the development of a web application, in order to improve the self descriptiveness of the web application.

In future work, [14] further experimentation aiming to assess the validity of the proposed approaches for improving the comprehensibility of a web application will be addressed.

2.2.8 Abdelwahab Hamou-Lhadj, Abdeslam En-Nouaary, Khalid Sultan: Maintaining a poorly documented web application is not an easy task; software engineers must understand various parts of an application before they can make changes that preserve reliability and other system attributes. In recent years, there has been a noticeable increase in the number of studies that aim at reverse engineering web applications. These studies embody a rich set of techniques that differ in a variety of ways. In this paper, we study several techniques for reverse engineering of web applications. The objective is to understand the trends in this area as well as uncover key research questions that remain unaddressed.

[16] proposed several techniques that vary depending on the levels of abstraction of the extracted views, the notation used to represent the content of the views, and the type of analyses performed. [16] presents the techniques covered in several studies. The main contributions of the [16] are twofold:

- Understand the current trends in reverse engineering of web applications.
- Discuss key research challenges that need to be addressed.

These techniques aim at recovery the design, architecture, user interface elements, and data from web applications. The objective is to help software engineers maintain web systems in an efficient manner. [16] also discussed several research challenges that need to be addressed such as dealing with the heterogeneous aspect of web systems, investigating techniques to cope with scalability issues when working on large applications, and finally fostering the use of dynamic analysis as a complementary view to the ones generated using static analysis.

Table 2.1 Summary of literature survey

Author Name & Year	**Description**
Byungjeong Lee, Eunjoo Lee, and ChisuWu	Describe the technique for web optimization by using genetic algorithm along with web page relationships and metrics(page similarity).
Filippo Ricca, Paolo Tonella , Ira D. Baxter	Restructuring web applications via transformation rules is trying to apply rewrite rules to web applications with the aim of restructuring them during the evolution phase.
Ching-Yi Cheo and Fun Ye	Describe the methodology for clustering using particle swarm optimization.
Xiaohui Cui, Thomas E. Potok, Paul Palathingal	Document clustering using the PSO algorithm for fast and high-quality document clustering algorithms .
G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini	Clustering method for decomposing a web application into groups of

Author Name & Year	**Description**
	functionally related components.
G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini	Tool for reverse engineering web applications.
G. A. Di Lucca, A. R. Fasolino, P. Tramontana	Proposed reverse engineering process includes five main steps: • Static analysis of the web application • Dynamic analysis of the web application • Automatic clustering of the web application . • Validation of the clustering • Abstraction of UML diagrams.
Abdelwahab Hamou-Lhadj, Abdeslam En-Nouaary, Khalid Sultan	Propose several techniques that vary depending on the levels of abstraction of the extracted views, the notation used to represent the content of the views, and the type of analyses performed.

Author Name & Year	**Description**
	The main contributions of the [16] are twofold: • Understand the current trends in reverse engineering of web applications. • Discuss key research challenges that need to be addressed.

Chapter 3

RESEARCH BACKGROUND

3.1 Key research concepts

Our research is based on the use of 'web application restructuring using Particle Swarm Optimization'. web restructuring is the main focus of this section for improving quality of the web application.

3.1.1 Analysis of software metrics

The Quantitative measurements are essential in all sciences; there is a continuous effort by computer science practitioners and theoreticians to bring similar approaches to software development.

Pressman explained as "A measure provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of the product or process".

Fenton defined measurement as "it is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules".

Software metrics can be defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products" [5].

Software metric is a measure of some property of a piece of software or its specifications. A metric is a measurement function, and a software quality metric is "a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality."

"A *software metric* is a quantitative measure of the degree to which a system, component or process possesses a given attribute" [5].

There are various challenges regarding software measurement like how to measure the size of software? How much will it cost to develop a software? How many bugs can we expect? When can we stop testing? When can we release the software?, What is the module strength and coupling ,etc. That's why we need software metrics.

3.1.2 Introduction to web metrics

Web metrics are used to better understand the attributes of the web page we create. But, most important, we use web metrics to assess the quality of the web engineered product or the process to build it. In web metrics we measure various things related to web pages like number of static web pages, number of dynamic web pages, number of internal page links, word count, web page similarity, web page search and retrieval, number of static content objects, number of dynamic content objects etc.

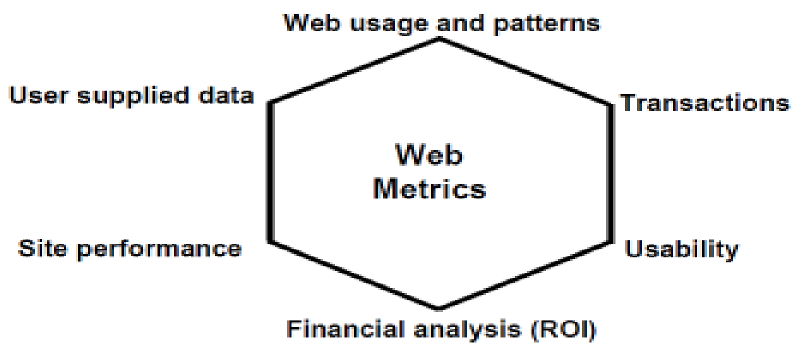

Figure 3.1: Web Metrics Components[7]

Since metrics are crucial source of information for decision making, a large number of web metrics have been proposed in the last decade to compare the structural quality of a web page. web metrics can be categories as following.

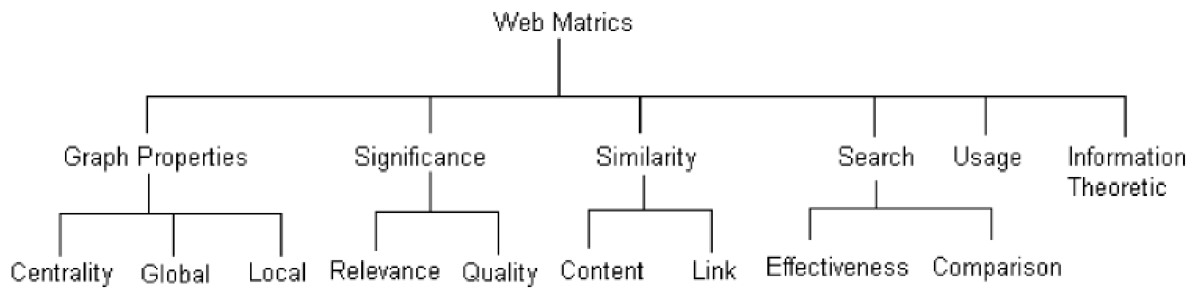

Figure 3.2: Taxonomy of Web Metrics[7]

3.1.2.1 Similarity Web Metrics:

Web page similarity metrics measure the extent of relatedness between two or more web pages. Similarity functions have mainly been described in the context of web page clustering schemes. Clustering is a natural way of semantically organizing information and abstracting important attributes of a collection of entities. Clustering has certain obvious advantages in improving information quality on the WWW. Clusters of web pages can provide more complete information on a topic than individual pages, especially in an exploratory environment where users are not aware of several pages of interest. Clusters partition the information space such that it becomes possible to treat them as singular units without regarding the details of their contents. Similarity metrics classified into i) Content based , ii) Usage based metrics ,ii) Link based

i). Content-based similarity metrics: Content-based similarity is measured by comparing the text of documents. Pages with similar content may be considered topically related and designated the same cluster.

ii) Usage-based similarity metrics: Usage based similarity is based on patterns of document access. The intent is to group pages or even users into meaningful clusters that can aid in better organization and accessibility of web sites.

iii) link-based similarity metrics: Link based measures rely exclusively on the hyperlink structure of a web graph to obtain related pages.

For grouping interacting various related pages, here, define link based page similarity (PS), based on the cosine equation, which is derived from page relationship weights.

$$PS(u,v) = \frac{\sum_{w \in WP} W(u,w) * W(v,w)}{\sqrt{\sum_{w \in WP} W(u,w)^2 * \sum_{w \in WP} W(v,w)^2}} \ .$$

[7]

where *WP* is a set of web pages.

We can also define cohesion and coupling metrics using page similarity based on connection strength considering the type and the number of the relationships and the number of parameters transferred between web pages.

$$COH(C) = \frac{\sum_{u,v \in C} PS(u,v)}{(|C| * |C|)} \ .$$

[1]

Where |C| is the number of pages in cluster C

Cohesion is a measure of how strongly related each piece of functionality expressed by the source code of a software module is or cohesion is a measure of how strongly-related or focused the responsibilities of a single module are,

$$COP(C_i, C_j) = \sum_{u \in C_i} \sum_{v \in C_j} PS(u, v) \ .$$ [1]

where |C| is the number of pages in cluster C

Clustering objective (*CO*) is defined to measure appropriateness of clustering using cohesion and coupling of web application derived from cluster metrics.

$$COH(WA) = \frac{\sum_{C_i \in WA} COH(C_i)}{NC} \ .$$ [1]

$$COP(WA) = \sum_{C_i, C_j \in WA} COP(C_i, C_j) \ .$$ [1]

$$CO(WA) = \frac{NC * (NC - 1) * w_{coh} * COH(WA)}{2 * w_{cop} * COP(WA)} \ .$$

In software engineering, coupling or dependency is the degree to which each program module relies on each one of the other modules. Coupling can be "low" (also "loose" and "weak") or "high" (also "tight" and "strong").

Coupling talks about the interdependencies between the various modules while cohesion describes how related functions within a module are. Low cohesion implies that module performs tasks which are not very related to each other and hence can create problems as the module becomes large.

3.2 Particle Swam Optimization

Particle swarm optimization technique was developed by Dr. Eberhart and Dr. Kennedy in 1995. It is a population based stochastic optimization technique. It has inspired by the social behavior of bird flocking or fish schooling.

About fish schooling –"in the theory at least individual member of the school can profit from the discoveries and previous experience of all the other members of the school during the search for the food" (a socio-biological E.O. Wilson)

Particle swarm optimization technique is based on social intelligence which exists in biological population. Social intelligence exhibits adaptive capabilities of people and animals by implementing an ''information sharing'' approach, furthermore also contributes to the creation, facilitation, and maintenance of critical behaviors.

PSO uses a population of particles for searching the optimum result. The population is called *swarm* and the individuals are called *particles.*

3.2.1 Introduction to Particle Swarm Optimization Algorithm

A particle swarm represents a bird flock and researches a solution in D-dimensional space. Each particle in the searching space has its own position vector, *Xi* .

The position of each particle is a possible solution and is calculated the particle's fitness by putting its position into objective function. The particle's next action is decided by velocity vector, *Vi*.

The PSO algorithm could be defined by the following equations (Shi et al. 2006)

$$V(k+1) = V(k) + c1 * \alpha * (P_{best} - X_{current}) + c2 * \beta * (G_{best} - X_{current})$$

$$X(k+1) = X(k) + V(k+1)$$

where P_{best} denotes the best position of the particle, $G_{best}$is the best position among all particles.c1 is the inertial weight factor, α and β is uniform random value in the interval [0, 1], *k* is the current generation number.

3.2.2 Block diagram of PSO Algorithm

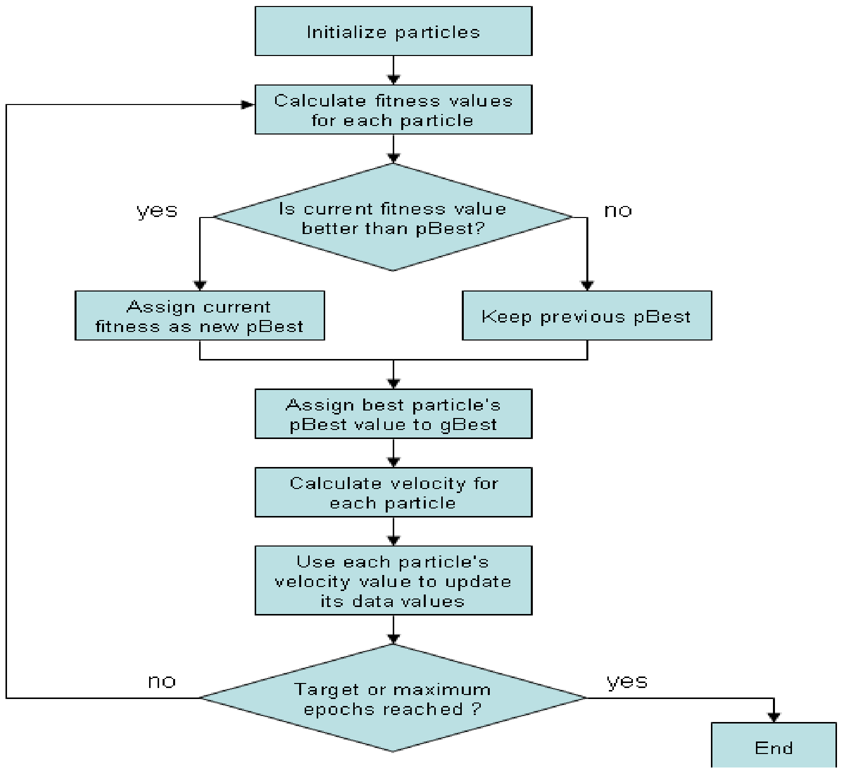
Initialize particles
Calculate fitness values for each particle
yes
Is current fitness value better than pBest?
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
no
Target or maximum epochs reached ?
yes
End

Figure 3.3: Block diagram of PSO algorithm

3.2.3 Strengths and Limitations

Particle swarm optimization has much property common with the genetic algorithm. The difference is that PSO does not have genetic operator like crossover and mutation. In PSO particles update themselves with help of its velocity.

3.2.3.1 Strengths:

- PSO starts from a population of individuals rather than a single individual thus are more likely to produce a global optimum solution.
- PSO the particles follow to the best particles only. So it is one way information sharing mechanism and looks for the best solution only.
- Compared to other evolutionary algorithm in PSO the particles try to converge to the best solution quickly even in local version in most cases.
- As compared to genetic algorithm it is easier to implement and have few parameters to adjust.

3.2.3.2 Limitations:

- The performance of the algorithm is highly dependent upon the derived fitness function.
- Deciding upon a fitness function is a challenging task. If the fitness function is not selected properly, the algorithm may not produce a global optimum solution.

3.3 Cohesion

Cohesion is a measure of how strongly-related each piece of functionality expressed by the source code of a software module is. Methods of measuring *cohesion* vary from qualitative measures classifying the source text being analyzed using a rubric with a hermeneutics approach to quantitative measures which examine textual characteristics of the source code to arrive at a numerical cohesion score. Cohesion is an ordinal type of measurement and is usually expressed as “high cohesion” or “low cohesion” when being discussed. Modules with high cohesion tend to be preferable because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability whereas low cohesion is associated with undesirable traits such as being difficult to maintain, difficult to test, difficult to reuse, and even difficult to understand.

Cohesion is often contrasted with coupling, a different concept. Nonetheless high cohesion often correlates with loose coupling, and vice versa.

3.3.1 Types of Cohesion:

Cohesion is a qualitative measure meaning that the source code text to be measured is examined using a rubric to determine a cohesion classification. The types of cohesion, in order of the worst to the best type, are as follows:

3.3.1.1 Coincidental Cohesion

Coincidental cohesion is when parts of a module are grouped arbitrarily; the only relationship between the parts is that they have been grouped together (e.g. a “Utilities” class).

3.3.1.2 Logical Cohesion

Logical cohesion is when parts of a module are grouped because they logically are categorized to do the same thing, even if they are different by nature (e.g. grouping all mouse and keyboard input handling routines).

3.3.1.3 Temporal Cohesion

Temporal cohesion is when parts of a module are grouped by when they are processed - the parts are processed at a particular time in program execution.

3.3.1.4 Procedural Cohesion

Procedural cohesion is when parts of a module are grouped because they always follow a certain sequence of execution (e.g. a function which checks file permissions and then opens the file).

3.3.1.5 Communicational Cohesion

Communicational cohesion is when parts of a module are grouped because they operate on the same data (e.g. a module which operates on the same record of information).

3.3.1.6 Sequential Cohesion

Sequential cohesion is when parts of a module are grouped because the output from one part is the input to another part like an assembly line (e.g. a function which reads data from a file and processes the data).

3.3.1.7 Functional Cohesion

Functional cohesion is when parts of a module are grouped because they all contribute to a single well defined task of the module (e.g. tokenizing a string of XML). Xohesion is a ranking type of scale, the ranks do not indicate a steady progression of improved cohesion. First two types of cohesion are inferior; communicational and sequential cohesion are very good; and functional cohesion is superior.

3.4 Coupling

In software engineering, *coupling or dependency* is the degree to which each program module relies on each one of the other modules.

Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa.

3.4.1 Types of Coupling

Coupling can be "low" (also "loose" and "weak") or "high" (also "tight" and "strong"). Some types of coupling, in order of highest to lowest coupling, are as follows:

3.4.1.1 Content Coupling

Content coupling (also known as Pathological coupling) is when one module modifies or relies on the internal workings of another module (e.g., accessing local data of another module). Therefore changing the way the second module produces data (location, type, timing) will lead to changing the dependent module.

3.4.1.2 Common Coupling

Common coupling (also known as *Global coupling*) is when two modules share the same global data (e.g., a global variable). Changing the shared resource implies changing all the modules using it**.**

3.4.1.3 External Coupling

External coupling occurs when two modules share an externally imposed data format, communication protocol, or device interface. This is basically related to the communication to external tools and devices.

3.4.1.4 Control Coupling

Control coupling is one module controlling the flow of another, by passing it information on what to do (e.g., passing a what-to-do flag).

3.4.1.5 Stamp Coupling (Data-structured coupling)

Stamp coupling is when modules share a composite data structure and use only a part of it, possibly a different part (e.g., passing a whole record to a function that only needs one field of it). This may lead to changing the way a module reads a record because a field that the module doesn't need has been modified.

3.4.1.6 Data Coupling

Data coupling is when modules share data through, for example, parameters. Each datum is an elementary piece, and these are the only data shared (e.g., passing an integer to a function that computes a square root).

3.4.1.7 Message Coupling

This is the loosest type of coupling. It can be achieved by state decentralization (as in objects) and component communication is done via parameters or message passing (see Message passing). No coupling modules do not communicate at all with one another.

3.4.1.8 Subclass Coupling

Describes the relationship between a child and its parent. The child is connected to its parent, but the parent isn't connected to the child.

3.4.1.9 Temporal Coupling

When two actions are bundled together into one module just because they happen to occur at the same time.

Chapter 4

Research Methodology

4.1 Particle Swarm Optimization Based Clustering

In this particle swarm optimization technique we will represent each solution by a particle. The position of each particle will be represented by an N-integer string, in which the value at the ith location is the identifier of the cluster of which ith webpage of the selected cluster is a member. This representation is shown in diagram below:

Webpage	1	2	3	4	5	6	7	8	9	10
Cluster	3	1	2	4	1	1	2	4	3	2

Figure 4.1 Initial clustering of pages

We define a fitness function to evaluate the fitness of each particle. This objective function is defined as:

$$FITNESS(WA) = 1/ CO(WA)$$

Here WA represents a web application with set of cluster and CO is *cluster objective*. Cluster objective is defined as measurement of appropriateness of cluster using cohesion and coupling.

In the next section we applied a hybrid PSO algorithm to find optimize clusters. In this hybrid particle swarm optimization algorithm we have used the concept of crossover operation in the PSO to make its search broader and randomize.

4.2 Proposed clustering algorithm

Find relationship in web application, compute *COH(WA)* and *COP(WA)* form the relationship.

Step 1: Generate population:

N number of particle is generated randomly. i.e. (P_1, P_2, P_3……..P_n)

Step 2: Initialization:

Position and velocity corresponding to each particle is generated randomly. Position and velocity of a particle helps in observing fitness of the particle and the best particle is selected from the population.

Step 3: Each particle updates it velocity and position with respect to the best particle. In this step we update the position and velocity if the new position and velocity has higher fitness value otherwise retain the previous value.

Step 4: Update population by applying crossover operation by selecting best particle each time.

Step 5: Find the fitness value of newly generated particles and select the fittest particle among all.

This process of hybrid PSO is repeated until the ending criteria does not meet. The ending criteria could be:

(1) Number of maximum defined iteration reached.

(2) Particles fitness does not improve further.

(3) A user defined fitness value is achieved by particles.

4.3 Case Study

Figure 4.2 shows contents of a small in form of UML package diagram. A package stands for a cluster to which pages belong. The number in page name is the unique identifier and the letter I in index.asp name of root package indicates the first page of the site.

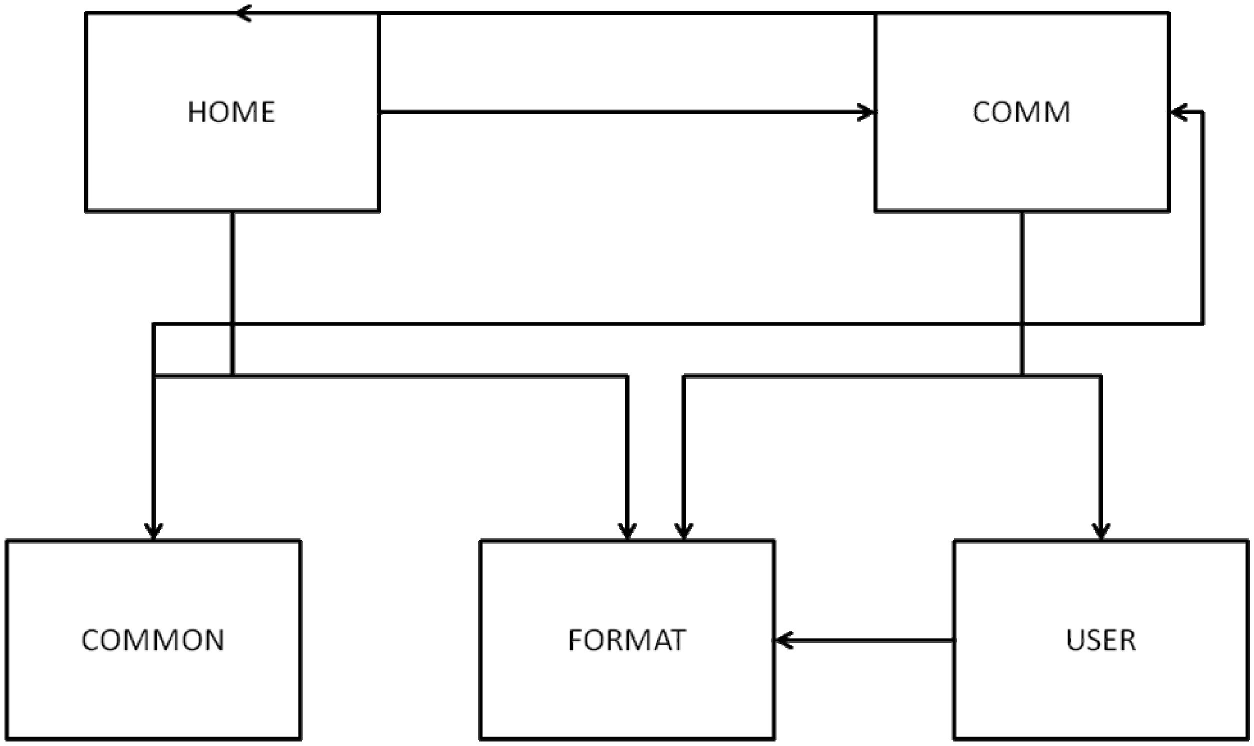

Fig 4.2: Structure of Web Application

Table 4.1 lists out the member pages of all groups in the web application we are examining in this study.

S.No.	Group	Member Pages
1	Home	index.php, table.php, main_nav.php
2	Comm	login.php, my_data.php, category.php
3	Common	footer.php, header.php
4	Format	style.css
5	User	index.php

Table 4.1: Members of Group in Web Application

For this proposed PSO algorithms, we choose 50 numbers of particles in the initial population and the uniform crossover rate is 0.5. That means after applying crossover, 50% of the old population will be replaced by new off springs. We will stop iterating when a termination criterion achieved. Termination criteria can be a predefined fitness value or fixed number of iterations. Figure 4.3 shows the navigation starting from the entry page. Two index.asp pages are displayed with path since there are three pages with the same name in different packages. Other pages are unique. Figure 4.3 shows the structure of a clustered Web application when wln, wsu, wre, win, wld, wcoh and wcop are 1, 2, 2, 1, 1, 0.6 and 0.4, respectively. Each polygon represents a cluster. The application has been grouped into six clusters which are one more than those in Fig. 4.2. The fitness function value has decreased and converged. The clustered web application in Fig. 4.3 can be refined.

[1] indicated that the structure of web applications clearly has something to do with quality attributes such as complexity or scalability as well as cohesion and coupling. We used their work as basis and used PSO instead of GA to find clusters of web pages with good cohesion and coupling. The results show that PSO is capable of forming clusters and the results can be verified by comparing them with the results showcased by [1].

Possible directions for future work are,

1. We need a solid ground to verify the superiority of PSO over GA in this domain and using those grounds we need to validate studies like these. Although PSO is superior to GA in almost all applications till date, still, we need to make sure that we have concrete proof of this fact when it comes to web applications. This would involve testing both algorithms over a diverse set of applications and examining the results of clustering.
2. In our work we only used quality attributes like coupling and cohesion to group the web pages. Modern software is examined over a large number of quality attributes and metrics and since modern web applications are no less than full functional software systems with some applications clearly out performing classic software in every department from size to scalability we need to include more quality attributes in such studies and experiments.

References

[1] Byungjeong Lee, Eunjoo Lee and ChisuWu , “Genetic Algorithm Based Restructuring of Web Applications Using Web Page Relationships and Metrics” School of Computer Science, University of Seoul, Korea Department of Computer Engineering, Kyungpook National University, Korea School of Computer Science and Engineering, Seoul National University, Korea.

[2] San Murugesan, Yogesh Deshpande, Steve Hansen and Athula Ginige, "Web Engineering: A New Discipline for Development of Web-based Systems," Proceedings of the First International Conference of Software Engineering (ICSE) Workshop on Web Engineering, Los Angeles, USA, 1999.

[3] What is a Web Application? (http:/ / webtrends. about. com/ od/ webapplications/ a/ web_application. htm) by Daniel Nations, former About.com Guide

[4] Elliot 1. Chikofsky, Index Technology Corp. and Northeastern University , James H. Cross I/, Auburn University “Reverse Engineering and design recovery: A taxonomy”.

[5] K.K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Empirical Study of Object- oriented Metrics", November –December 2006.

[6] Software Metrics: Successes, Failures and New Directions Norman E Fenton and Martin Neil Centre for Software Reliability, City University Northampton Square, London EC1V OHB , United Kingdom.

[7] 'A Survey of Web Metrics' , DEVANSHU DHYANI, WEE KEONG NG, AND SOURAV S. BHOWMICK ,Nanyang Technological University.

[8] 'Particle Swarm Optimization Algorithm and Its Application to Clustering Analysis' Ching-Yi Cheo and Fun Ye IEEE Taiwi. Taiwan. March 21-23, 2004 lntemationai Conference on Networking, Sensing Control.

[9] 'Document Clustering using Particle Swarm Optimization' Xiaohui Cui, Thomas E. Potok, Paul Palathingal , 2005 IEEE.

[10] 'Comprehending Web Applications by a Clustering Based Approach' G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini. 10 th International Workshop on Program Comprehension (IWPC'02) , 2002 IEEE.

[11] V. Tzerpos, R.C. Holt, “On the stability of software clustering algorithms”, *8th* International Workshop on Program Comprehension*,* IEEE CS Press, Los Alamitos, CA, 2000, pp. 211-220.

[12] V. Tzerpos, R.C. Holt, “MoJo: a distance metric for software clusterings”, 6th Working Conference on Reverse Engineering, IEEE CS Press, Los Alamitos, CA, 1999*,* pp. 187-193.

[13] G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini “WARE: a tool for the Reverse Engineering of Web Applications” Sixth European Conference on Software Maintenance and Reengineering (CSMR.02) , 2002 IEEE.

[14] G. A. Di Lucca, A. R. Fasolino, P. Tramontana “Towards a Better Comprehensibility of Web Applications: Lessons Learned from Reverse Engineering Experiments” Fourth International Workshop on Web Site Evolution (WSE’02) , 2002 IEEE.

[15] G. A. Di Lucca, M. Di Penta, G.Antoniol, G.Casazza, “An approach for reverse engineering of web-based applications”, Proc. of 8th Working Conference on Reverse Engineering, IEEE CS Press, Los Alamitos, CA*,* 2001, pp. 231-240.

[16] Abdelwahab Hamou-Lhadj, Abdeslam En-Nouaary, Khalid Sultan “Reverse Engineering of Web Based Systems” 2008 IEEE.