IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-7, Issue-3, March 2019 E-ISSN: 2347-2693

A Systematic Review of Feature Location Techniques under Software Change Impact Analysis

Ankit Dhamija[1*], Sunil Sikka[2]

[1,2]School of Engineering & Technology, Amity University Haryana, Gurgaon, India

[*]*Corresponding Author: adhamija@ggn.amity.edu, Tel.: 9729658806*

DOI: https://doi.org/10.26438/ijcse/v7i3.184192 | Available online at: www.ijcseonline.org

Accepted: 14/Mar/2019, Published: 31/Mar/2019

***Abstract*—** The possibility of introduction of a change in software cannot be denied as the request for upgrades and improved functionality keeps coming on. Implementing these changes require a systematic Change Impact Analysis (CIA) which is a step by step process under software maintenance. However, the most difficult phase in this systematic CIA process is the identification of an initial location of initiating the proposed change. Various techniques have been proposed to identify this initial location which comes under Feature Location Techniques. These techniques are aimed at finding areas in the software code and other software artifacts that implement a feature. The paper attempts to organize and structure existing work in the field of feature location by presenting a literature survey of recent feature location techniques whereby the techniques have been categorized according to the methodology followed, the tools proposed and their impact. The paper also discusses open issues and defines future directions in the field of feature location.

***Keywords*—**change impact analysis, feature location, software maintenance,concept location

I. INTRODUCTION

Software maintenance is the most costly process in the entire software life cycle [1] where modifications to the software keep on coming due to various reasons like error correction, adapting to a different environment and for adding new functionality. The major effort in software maintenance goes into the bringing change in the software code, the way different software components interact and its corresponding software architecture. Analyzing these changes, implementing the necessary ones and ensuring the consistency of software with other interacting entities is a very challenging task. To make sure that the software functions as desired, even after the introduction of changes, a systematic and careful Software Change Impact Analysis (CIA) is required, so that effects of the change can be predicted and accordingly, further actions can be initiated. It includes a variety of techniques that are used for finding out the possible effects a change may bring on the introduction of a proposed change on other software elements. It can be used for predicting change impacts, making cost estimates and for program understanding before implementing a proposed change. It can also be used for analyzing the adverse & ripple effects on other software elements, for selecting test cases and for performing change propagation, after change implementation. Several definitions have been proposed by earlier researchers. The most widely recognized and quoted definition was proposed by Bohner et al [2] in 1996, where CIA is defined as "*the process of identifying the potential consequences of a change, or estimate what needs to be modified to accomplish a change*". Prior to them, Pfleeger et al [3], in 1990 defined CIA as *"the evaluation of the many risks associated with the change, including estimates of the effects on resources, effort, and the schedule"*. Before that, in 1986, Horowitz *et al* [4] defined CIA as *"an examination of an impact to determine its parts or elements"*. Thus over the past 30 years, so many definitions and techniques have been proposed by earlier researchers.

The first step in CIA is to identify an initial location in the source code that corresponds to a specific functionality, which is known as *feature* (or *concept*) *location*. Wilde & Scully [5] defined feature location as "the activity of identifying the source code elements (i.e., methods) that implement a feature." It can be considered as a process of finding out the starting location in the code implementing some functionality or feature. A lot of information regarding this can be found from the comments written by the code developers; and also from the identifiers used in the code. Such information may reveal a lot about the design of a software system but this information is available in unstructured form. Therefore techniques are proposed to identify the relevant areas in the source code that implements a feature. It is one of the most frequent maintenance

activities undertaken by developers because it is a part of the incremental change process [6]. During the incremental change process, programmers use feature location to find where in the code the first change to complete a task needs to be made. The full extent of the change is then handled by impact analysis, which starts with the source code identified by feature location and finds all the code affected by the change. Feature location is one of the most important and common activities performed by programmers during software maintenance and evolution. No maintenance activity can be completed without first locating the code that is relevant to the task at hand, making feature location essential to software maintenance. As the change request comes to the maintenance team, they are supposed to find out the areas in the software code to initiate a change. However, due to lack of documentation, the maintenance team is forced to perform a manual search using pattern matching techniques which is very time consuming. Thus, the researchers in academics and industry have been proposing various Feature Location techniques which are unique in every sense- their input requirements and the algorithms they use.

In this paper, a comprehensive literature review of the recent innovations in feature location techniques (FLT) is presented. Section II presents the systematic CIA process, section III presents the research methodology, section IV presents the detailed literature review of advancements in feature location techniques, section V performs the comparative analysis of the various FLT, section VI presents the open research areas in feature location and section VII concludes the paper.

II. SYSTEMATIC CIA PROCESS

It is very seldom that software has been designed, developed, tested well and delivered to the client without any requests for modifications. The requests to developers for change in software are natural and every new release of software is bound to have some or the other change in its functionality. However, these change requests must be carefully understood and the impact that it will make on software and how the other associated parts of software will be affected must be thoroughly analyzed by the development team before proceeding towards implementing the change requests. In short, a careful and systematic Change Impact Analysis is required. The most widely used systematic CIA process is presented in figure 1 [7].

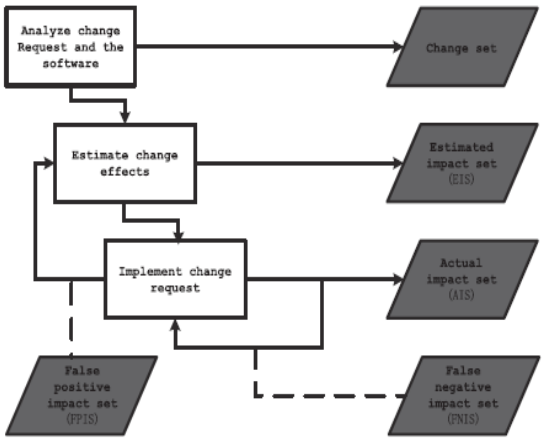

Figure 1: Systematic CIA Process[7]

STEP 1: **Analyze both, the software and the change request.**

CIA process begins by analysis of change requests and identification of areas in source code that may be affected due to introduction of change. Here, a set called *Change Set* is prepared which includes the tentative impacted areas due to change introduction. This identification of an initial location in source code is called feature location [8].

STEP 2: **Estimate change effects**

In the second step, impact of the changes on other elements in the software are estimated using various change impact analysis techniques and a set known as *Estimated Impact Set (EIS)* is created.

STEP 3: **Implement change request**

Here, the actual implementation of changes is done to fulfil the client's request and the components are actually modified in the *Actual Impact Set (AIS)*. Thus, AIS includes areas where the changes have been actually implemented.

Two parameters are generally used to judge the CIA effectiveness:

False Negative Impact Set (FNIS)

FNIS is a set of number of extra elements impacted by CIA process that were not estimated by EIS. It means that while change implementation, developers may come across some areas in software that were initially not considered to be part of EIS but not during change implementation, they are being discovered. Thus FNIS denotes an under estimation of impacts.

False Positive Impact Set (FPIS)

FPIS is a set of number of extra elements included in EIS that don't require any modifications.

It means that while change implementation, developers realize that some areas that were considered as requiring

changes don't need to be modified at all. Thus, FPIS denotes the over-estimation of impacts.

The ultimate objective of carrying out CIA is to make sure that the difference between EIS and AIS is minimized. The relationship among all these sets can be represented as:

$$(EIS+FNIS)-FPIS=AIS \quad \text{Eq (1)}$$

The above equation is the expected and best case scenario but it's seldom achieved. This is because of the reason that identifying the various sets like change set, EIS; AIS etc require the selection of most appropriate CIA techniques and tools. Thus, it can only be achieved through proper understanding of the software, various types of CIA techniques available and a judgment about the best possible combination of CIA techniques for a particular project. [26] Presented an analysis of various software metrics for object oriented development where they provided a basis for measuring all of the characteristics like size, complexity, performance and quality.

III. METHODOLOGY

Following research questions are formulated to obtain the current state of Feature Location Techniques:

- **RQ1**: What are the recent advancements and trends in the field of Feature Location?
- **RQ2**: Which properties can be identified from the recent feature location techniques?
- **RQ3**: What are the future areas of research in feature location?

Figure 2 presents the approach planned to answer the above research questions:

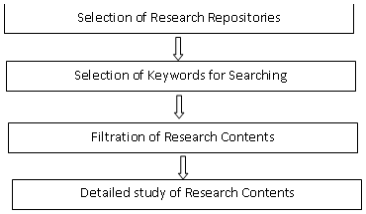

Figure 2: Methodology for Paper Selection

1. **Literature Survey**: This is carried out through extensive search in electronic databases like IEEE Computer Society, ACM, Elsevier, Springer etc.

2. **Keywords Search**: In the second stage, various appropriate keywords were used to search the required research contents.

3. **Paper Filtration**: In this stage, the papers were filtered based on the paper title and its relevancy and the abstract.

4. **Detailed Study**: In fourth stage, the papers filtered were studied in full depth.

Following the above mentioned methodology, a total of 46 research papers were found from various research repositories and after proper filtration, 18 papers were considered into consideration for the review. Table 1 presents the data about the research paper collection.

Table 1: Paper Selection Statistics

Database	"Change Impact Analysis" + "Feature Location"
IEEEXplore	16
ACM Digital Library	0
Springer	14
Elsevier	13
Others	3
Total	46
Suitable papers after Filtration Process	**18**

From the selected papers, some basic information is extracted as shown in figure 3.

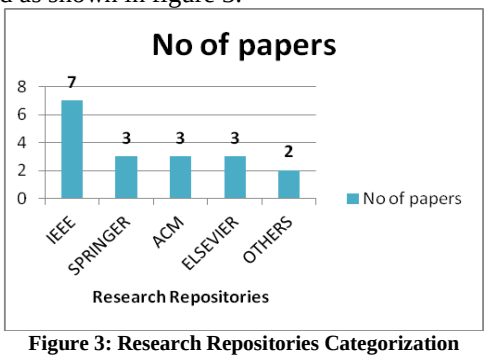

Figure 3: Research Repositories Categorization

IV. RELATED WORK

In this section, the researcher will review the various Feature Location Techniques proposed by fellow researchers till now and a careful analysis of all the techniques on the following parameters is performed to judge the viability of the techniques:

1. Technique type
2. Experimental / Empirical or Case Based
3. Tool Support

As it is very difficult and time consuming process to fetch each research paper from the repositories, therefore, during search, the researcher also took into consideration the review papers based on feature location. A total of 04 review papers were found which presented the complete review of feature location techniques up to the year in which these review papers got published. The following are details of these papers:

Table 2: Summary of Review Papers based on FLT

Year & Reference	Description	Techniques Discussed	Results

2009, M.Revelle et al [9] ICPC, IEEE	Presents exploratory study of 10 FLT's performing performs textual, dynamic and static analysis in various combinations. Evaluates approaches to find several relevant methods. A new approach involving textual analysis is proposed.	1. Latent Semantic Indexing (LSI) based on Textual Analysis 2. Other Textual Analysis techniques: Information Retrieval, independent component analysis and Natural Language 3. Dynamic Analysis using scenarios and collecting traces including full traces and marked traces. Others include software reconnaissance, SPR to rank methods 4. Static Analysis using Program Dependency Graphs (PDG) 5. Textual analysis using nl-queries and method queries. 6. Hybrid approaches: CERBERUS	Method-queries perform better than human formed queries.
2011 Dit et al [8] CRC to Journal of Software Maintenance and Evolution: Research and Practice	Presents a systematic survey of 89 articles and classify them according to their category. Analyze them according to seven dimensions where each dimension has its associated set of attributes. Also presents issues and future directions in feature location CIA.	*1. Dynamic Techniques:* software reconnaissance, Dynamic Feature Traces (DFT), execution slices based, feature component maps, applying data mining on execution traces, minimally intrusive instrumentation technique called *minist.* *2. Static Techniques:* Abstract System Dependence Graphs (ASDG), Concern Graphs Representation, FRAN (Finding with RANdom walks), topology analysis, static dataflow analysis *3. Textual Techniques:* grep pattern matching, Formal Concept Analysis (FCA), information retrieval, Independent Component Analysis , NLP, Action-Oriented Identifier Graph (AOIG), Resource Description Framework Graphs (RDF)	• Increased Overheads in using dynamic techniques. • Results produced by Textual techniques are dependent on the quality of the queries used. • Mixed approaches are being proposed more by researchers. • Limited comparisons between existing and new approaches and limited benchmarks for comparison.
2013 J Rubin et al [10] SPRINGER	Focus on automated FLT's and describes the implementation strategies of existing FLT's. Presents their pros & cons and provides circumstances where each technique is to be used.	1. Term frequency inverse document frequency (tf-idf) 2. Program dependence analysis (PDA)Formal Concept Analysis 3. Latent Semantic Indexing (LSI) 4. Hyper-link Induced Topic Search (HITS)	• No single technique is best fit for all cases. • Analyzed techniques on several criteria: strongly coupled implementation, meaningful names, and change history and execution scenarios.
2013 N Alhindawi et al [11] JSEA SciRes	Provides review of enhancement techniques related to feature location and emphasizes its role in and maintenance & program comprehension.	1. Same techniques as in above review papers.	Scope of improvement in the accuracy of methods proposed by earlier researchers.

Out of the four review papers, the researcher found that two of them have done very exhaustive literature survey of existing FLT's and have tried to give a comprehensive analysis of the techniques. However, the conclusion that can be deduced from all these review papers is that no single approach or technique can be declared to be the best one or most suitable for all software projects; Thus, all of them have asserted that a proper mix of static, dynamic and textual and historical techniques is needed to perform Feature Location tasks.

The above survey papers covered publications up to year 2013 and therefore, in the below section, we will discuss new and innovative FLT's proposed during 2014 till early 2017. The papers referred are from repositories including IEEE Computer Society, ACM, Springer and Elsevier.

E. Hill et al [12] proposed the use of positional proximity within natural language phrases (NL) using ad hoc considerations and Markov Random Field (MRF) modelling which they applied in five open source Java Systems to identify over 200 features. Their results indicated that larger variations exist for the ad-hoc positional-proximity based approaches than based on MRF based approaches.

F.Beck et al [13] introduced a fresh GUI for feature location and implemented as an Eclipse Plugin, *called In Situ Impact Insight* (I3), which assisted developers in exploring and examining the fetched code elements. It allows Developers in fetching textual similarity details of a code entity to the search query and also the valuable information on co-changed elements from a project's history. In their paper, they performed three feature location tasks from jEdit open source software and results showed the intuitiveness of I3 and proved that it provides developers the much needed information whenever they need it.

C.S. Corley et al [14] stated that Text retrieval based FLT's needs to be trained on source code snapshots which may lead to model obsolescence and latest code snapshot needs to be retained. Based on this need, they proposed FLT based on *topic modeling* where source code history is used to build the

model incrementally. They showed that the accuracy of a changeset-based FLT is similar to that of a snapshot-based FLT, but without the retraining costs. In another paper [15], they proposed a technique called Document Vectors (DV's) based on Deep Learning Models which according to them, works well with source code as they both capture the influence of context on each term in a corpus and map terms into a continuous semantic space that encodes semantic relationships such as synonymy. Their results show that DV's based FLT is better than Latent Dirichlet allocation (LDA) based FLT.

M Chochlov et al [16] proposed the use of Changeset Descriptions as a Data Source to assist Feature Location. They stated that existing textual FLT's based on Information Retrieval depend on code comments and identifiers to provide description of software entities. As an alternative to this, they proposed the use of changeset descriptions of the modified code in that changeset as a data source to describe software entities. During empirical evaluation of the proposed IR technique, their technique was found efficient in terms of effort.

G Liang et al [17] advocated the use of automatic FLT's to enable developers to quickly locate program segments. The existing FLT's had limitations like less availability of detailed sources, less analysis of internal behaviors and ineffectiveness for service-relevant code entries identification. Thus, they proposed a FLT based on behavior model and implemented a tool *BMLocator* that works in both online and offline mode and uses Natural Language and code analysis in a static manner to retrieve the of code behavior models. When a service description is provided, the tool in the first step fetches the behavior model and in second step, service relevant code units are suggested.

B.Dit et al [18] did an exploratory study on two FLT's founded on Information Retrieval and another one in combination of IR with dynamic analysis. These two techniques utilized three strategies for splitting identifiers to find out the best technique for splitting identifiers to improve accuracy. Their results show that FLT using IR can be effective in some cases but the other technique's results were not that significant.

J Burke et al [19] proposed ways to reuse evolve and utilize the feature location and enhancement techniques.

X Peng et al [21] presented an iterative approach with an assumption that features are inter-related and the program code is a resemblance of the features. The proposed approach considers the structural similarity between a program element and feature to ascertain significance and relation between features and elements. Then it uses iterations for distributing the importance of the feature-element mappings to the close by features and program elements. Results showed that their approach is robust, increase in the recall and a minor decrease of precision.

T Eisenbarth et al [20] presents a static-dynamic combinatorial and semi automatic approach that reconstructs feature mapping and exhibit and observable behavior. The proposed technique distinguishes between general and specific computational units with respect to given set of features and preserve the mental map of the analyst.

M D Almeida Maia et al [22] analyzed and presented the results of an experiment emphasizing how the execution traces can be helpful in maintenance tasks. Maintenance activity time and greater accuracy were the outcomes of the experiment that doesn't seem quite useful in maintenance.

S Zamani et al [23] proposed text retrieval based approach called term weighing that was used for adjusting a term importance within a document. Their work followed a noun based approach that considers the frequency of a term being used in the repositories. Their results show that the time usage in the weighting of terms, along with the usage of noun terms, improves a FLT relying on textual information.

G Scanniello et al [24] tried to improve text based feature location problem by utilizing dependencies between source code elements. Link analysis algorithm and Dependency information was used for ranking the document space and improving the text retrieval based feature location. To implement the approach, PageRank algorithm was used which is also used for retrieving applications related to web document retrieval. Their results showed better retrieval performance than the previous ones.

K Damevski et al [25] performed a field study about the way feature location within source code by developers, based on two data sets. Their results suggested the urgency of proposing more improved code search queries and there is a lack of code search tools to handle both lookup and exploratory queries. [27] proposed a tool for requirements management in object oriented systems for maintaining the status of requirements. As the requirements can also be helpful in feature location, this tool can be very useful in identifying an initial location of initiating a change.
The above information is summarized in the table below:

Table 3: Summary of Recent Research Papers based on FLT

Year &	Reference	Description of Work Done	Technique/Tool	Type of Project	Areas Focused	Results

Publisher			**Proposed**	**worked**		
2014, IEEE	E Hill et al [12]	Used Positional proximity of words in source code files	Positional Proximity using Natural Language based on Ad-hoc and MRF modeling	Open Source Java Systems	Source code search, Software Maintenance	Consistent results with MRF modeling than Ad-hoc
2015, IEEE	F Beck et al [13]	Introduced a user interface to support feature location	*In Situ Impact Insight* (I3) tool	Implemented as eclipse plugin, on jEdit issue tracking system	Textual similarity, execution traces, code inspection	Intuitive user interface
2015, IEEE	C S Corley et al [14] [15]	Introduced a topic-modeling based FLT in which the model is built incrementally from source code history.	Topic modeling based FLT	Open Source Java Projects	Text retrieval models, topic modelling	Same accuracy as snapshot-based FLT, No retraining costs.
		Use of a particular deep learning model, document vectors (DVs) in source code, For feature location.	Document Vector technique based on Deep Learning Models	Six software systems: ArgoUML, JabRef, jEdit, muCommander, Teaser, Python v2.7	Natural Language, Deep Learning, Document vectors	Better than FLT based on latent Dirichlet allocation (LDA)
2015, IEEE	M Chochlov et al [16]	Employing the changeset descriptions of the code altered in that changeset as a data source to describe software entities.	Technique using changeset descriptions and performed empirical evaluation to observe performance	Rhino and Mylyn.Tasks systems	IR, Software Repositories	Less effort required to configure the technique at method level granularity.
2016, IEEE	G Liang et al [17]	Behavior model based FLT applying static code analysis using NLP techniques in combination.	BMLocator Tool	Open Source: Tomcat and Hadoop	Behaviour Models, Natural Language Processing	Effective than existing techniques like TopicXP, CVSSearch**.**
2012, ACM	B.Dit et al [18]	Exploratory study. Two FLT's taking benefit of three strategies for splitting identifiers: CamelCase, Samurai and manual splitting of identifiers.	IR based and IR and Dynamic analysis based	Open source: Rhino and jEdit	IR, dynamic analysis	Effectiveness improved while using manual splitting.
2014, ACM	J.Burke et al [19]	addresses the feature location gap to improve feature addition and enhancements	No	NA	Feature Addition, Feature Enhancement, Feature Location	NA
2003, ACM	T.Eisenbarth et al [20]	Static- dynamic combinatorial approach that reconstructs feature mapping and exhibit and observable behavior.	--	Bauhaus	Program comprehension, formal concept analysis	distinguishes between general and specific computational units
2013,	X.Peng et al	iterative context-aware	iterative context-aware	DirectBank and	IR, structural	Robust, Recall

Elsevier	[21]	approach that takes into consideration the structural resemblance between a feature and a program element to determine feature-element relevance.	approach	Linux Kernel	similarity	may be increased and Minor decrease of Precision
2012, Elsevier	M.d. Almeida Maia et al [22]	Presented the results of an experiment emphasizing how the execution traces can be helpful in maintenance tasks.	Experimental	JHotDraw and ArgoUML	empirical assessment, execution traces	Greater accuracy, reduced maintenance activity time
2014, Elsevier	S Zamani et al [23]	text retrieval based approach called term weighing, used to adjust the term importance within a document.	new term weighting technique	Four open source projects	Noun usage, term weighting	Improved accuracy, effectiveness and performance
2015, Springer	G Scanniello et al [24]	Improvement in text based feature location by leveraging dependencies between source code elements	PageRank algorithm	jEdit and aTunes	Text retrieval	Better retrieval performance
2016, Springer	K Damevski et al [25]	a field study about the way feature location within source code is carried out by developers.	NO	Experimental		Suggestions for better code search queries/tools etc

V. FINDINGS & RESULTS

The following are the findings from the literature review:

a. The review on feature location techniques indicates that techniques based on Information Retrieval, Natural Language based and source code based are used by various researchers for carrying out the CIA task.

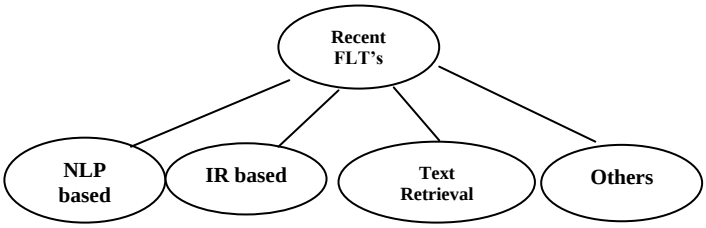

Figure 4: Paper Categorization

b. Moreover, the experimentation for all these approaches are being performed majorly on Open Source Systems which are mostly JAVA based systems.

c. The researchers have preferred open source software Systems like jHotDraw, Rhino, jEdit and others to test their proposed techniques.

d. Very few researchers have provided tool support for their techniques and most of the researchers have just provided their techniques and validated those using existing tools. Thus, there exists a gap where the authors should provide tool support along with their techniques because it is seldom that the existing tools prove to be successful with new techniques.

Table 4: Paper Type and Outcome

Technique & Reference	Type of Paper		Outcome	
	Review	Regular	Tool	Technique
T1 M.Revelle et al [9]	*			
T2 Dit et al [8]	*			
T3 J Rubin et al [10]	*			
T4 N Alhindawi et al [11]	*			
T5 E. Hill et al [12]		*		
T6 F.Beck et al [13]		*	*	*
T7 C.S. Corley et al [14]		*		*
T8 C.S. Corley et al [15]		*		*
T9 M. Chocolov et al [16]		*		*
T10 G Liang et al [17]		*	*	*
T11 B.Dit et al [18]		*		*
T12 J.Burke et al [19]		*		*
T13 T. Eisenbarth et al [20]		*		*
T14 X.Peng et al [21]		*		*

T15 M.d. Almeida Maia et al [22]	*	*
T16 S Zamani et al [23]	*	*
T17 G Scanniello et al [24]	*	*
T18 K Damevski et al [25]	*	*

e. There exists a scope of research area where the feature location techniques are applied and proposed on other than JAVA systems like web based systems, systems based on other languages like PHP and Dot Net.

f. Findings suggest that the documentation at various stages of software design and development plays a major role in identifying the initial location to implement a change. Various types of documentation like source code history, internal code behavior, execution traces, dependency analysis among software artifacts and code search queries are very helpful and widely used while creating a FLT.

VI. CONCLUSION AND FUTURE SCOPE

In this paper, the authors bring to fore the latest research developments in the area of feature location under change impact analysis. A systematic CIA process is also presented. During the review, the authors analyzed the recent review papers and research papers in detail and tried to categorize the results according to various criteria. Findings reveal that textual, historic information and documentation about the various stages of software development are used in the proposed techniques to identify the initial location of implementing a change. The proposed techniques use innovative methods based on natural language processing, information retrieval and text retrieval approaches. The authors are hopeful that the analysis carried out in this study will help the researchers in proposing more innovative techniques in the near future.

REFERENCES

[1] W.Li, S.Henry, *"Maintenance support for object-oriented programs"* Vol **7,** No 2, pp. 131–147, 1995.

[2] S Bohner, R. Arnold, *"Software Change Impact Analysis*", IEEE Computer Society Press: Los Alamitos, CA, USA, 1996.

[3] S.L. Pfleeger, S.A.Bohner, *"A framework for software maintenance metrics*", In the Proceedings of the International Conference on Software Maintenance, Washington, DC, pp. 320–327, 1990.

[4] E. Horowitz, R.C. Williamson,*"SODOS: a software documentation support environment—its definition*", IEEE Transactions on Software Engineering, Vol 12, No 8, pp. 849–859. 1986.

[5] N.Wilde, M.Scully,*"Software Reconnaissance: Mapping Program Features to Code*", Software Maintenance: Research and Practice, vol. 7, pp. 49-62, 1995.

[6] V. Rajlich and P. Gosavi,), *"Incremental Change in Object-Oriented Programming"*, IEEE Software, pp. 2-9. 2004.

[7] A.Dhamija, S.Sikka, *"Software Change Management: A Quantified Perspective"*, International Journal of Engineering & Technology-UAE, Vol 7, Issue 3.12, pp. 963-967. 2018.

[8] B.Dit, M.Revelle, M.Gethers, D.Poshyvanyk, *"Feature location in source code: a taxonomy and survey*", Journal of Software Maintenance and Evolution: Research and Practice, 2011.

[9] M. Revelle and D. Poshyvanyk, *"An Exploratory Study on Assessing Feature Location Techniques",* In Proceedings of 17th IEEE International Conference on Program Comprehension (ICPC'09), Vancouver, British Columbia, Canada, May 17-19, pp. 218-222, 2009.

[10] J. Rubin and M. Chechik, "A survey of feature location techniques," Domain Engineering: Product Lines, Conceptual Models, and Languages. Springer, pp. 29–58, 2013.

[11] N. Alhindawi, J. Alsakran, A. Rodan, H. Faris, *"A Survey of Concepts Location Enhancement for Program Comprehension and Maintenance*", Journal of Software Engineering and Applications, Vol 7, pp. 413-421, 2014.

[12] E. Hill, B. Sisman, A.C. Kak, *"On the use of positional proximity in IR-based feature location*", CSMR-WCRE,pp. 318–322, 2014.

[13] F. Beck, B. Dit, J. Velasco-Madden, D. Weiskopf, and D. Poshyvanyk. Rethinking user interfaces for feature location. In Proceedings of the 23rd IEEE International Conference on Program Comprehension, ICPC, pages 151–162. IEEE, 2015

[14] C.S. Corley, K.L. Kashuda, N.A. Kraft, *"Modeling changeset topics for feature location"* , *ICSME, Germany, IEEE.*, pp. 71-80, 2015.

[15] *C.S. Corley, K Damevski and N.A. Kraft*, Exploring the Use of Deep Learning for Feature Location, , *ICSME, Germany, IEEE*, 2015.

[16] M. Chochlov, M. English and J. Buckley, *"Using Changeset Descriptions as a Data Source to Assist Feature Location*",IEEE SCAM, Breman Germany, 2015.

[17] G Liang, Y Dang, H Chen, L Mei, S Li, Y M Chee, *"What Code Implements Such Service? A Behavior Model Based Feature Location Approach*", IEEE International Conference on Services Computing, 2016.

[18] B.Dit, L.Guerrouj, D.Poshyvanyk and G.Antoniol,*"Can Better Identifier Splitting Techniques Help Feature Location?"* In Proceedings. of 19th IEEE International Conference on Program Comprehension (ICPC'11), Kingston, Ontario, Canada, June 22-24 pp. 11-20, 2011.

[19] J.T. Burke, *"Utilizing Feature Location Techniques for Feature Addition and Feature Enhancement*", In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering pp. 879-882, 2014.

[20] T. Eisenbarth, R. Koschke and D. Simon, *"Locating Features in Source Code"* , IEEE Transactions on Software Engineering vol. 29 no. 3, pp. 210 – 224, 2003.

[21] X.Peng, X.Zhenchang, T.Xi, Yijun and Zhao, Wenyun). *"Improving feature location using structural similarity and iterative graph mapping*", Journal of Systems and Software, 86(3) pp. 664–676, 2013.

[22] M.D.A.Maia, R.F. Lafetá, *"On the impact of trace-based feature location in the performance of software maintainers*", Journal of Systems and Software, v.86 n.4, p.1023-1037, April, 2013.

[23] S. Zamani, S.P. Lee, R.Shokripour, J.Anvik J, *"A noun-based approach to feature location using time-aware term-weighting*". Inf Softw Technol, Vol 56, No 8, pp. 991-1011, 2014.

[24] G.Scanniello, A. Marcus, D. Pascale, *"Link analysis algorithms for static concept location: an empirical assessment*". Empirical Softw Eng, pp. 1–55, 2014.

[25] K. Damevski D. Shepherd L. Pollock "A field study of how developers locate features in source code" Empirical Software Engineering, pp. 1-24, 2015.
[26] Aanchal, S. kumar, "*Metrics for Software Components in Object Oriented Environments: A Survey*", International Journal of Scientific Research in Computer Science and Engineering, Vol.1, Issue.2, pp.25-29, 2013.
[27] Anandi Mahajan, Pankaj Sharma, "*Object Oriented Requirement management Tools for maintaining of status of requirements*", International Journal of Scientific Research in Computer Science and Engineering, Vol.6, Issue.6, pp.27-30, 2018

Authors Profile

*Mr. Ankit Dhamija pursued Master of Computer Applications from Kurukshetra University Kurukshetra in 2008*He is currently pursuing Ph.D. and currently working as Assistant Professor in Amity Business School, Amity University Haryana since 2012. He has published more than 15 research papers in reputed international journals including Scopus and conferences including IEEE and it's also available online. His main research work focuses on Software Maintenance, Cloud Security and Privacy, Management Informtion System. He has 10 years of teaching experience and 4 years of Research Experience.

Dr Sunil Sikka completeed his Ph.D from Maharishi Dayanand University, Rohtak. and currently working as Associate Professor in Amity School of Engineering & Technology at Amity University Haryana since 2014. He has published more than 20 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. His main research work focuses on Software Engineering, Spftware Testing, Data Mining, IoT and Computational Intelligence based education. He has 12 years of teaching experience and 8 years of Research Experience.

