Mapping and Visualization of Source Code: A Survey

Nakul Sharma, Prasanth Yalla

Department of Computer Science and Engineering, Koneru Laxmiah Education Foundation, Vijayawada, India.

e-mail: nakul777@gmail.com; prasanthyalla@kluniversity.in

***Abstract* - This paper summarizes various states of art literature in the field of mapping and visualizations in a programming language. The main focus is the design and analysis phase of Software Development Life Cycle. Mapping and visualization in software artifact is essential component for better understanding of the software. It also helps developers and maintenance professionals to visualize the effect of changes conducted in source code. In this work, a summary of literature review in of major tools developed for visualization and mapping of source code is provided.**

Keywords - Source Code Analysis, Mapping, Visualization, Source Code Visualization, Software Comprehension

I. INTRODUCTION

Mapping and visualization of the software are useful for the developer. In the object-oriented classes, the developer gets help from the visual aspects of how data gets passed from one class to another. In the case of procedural oriented programming as well, how the data is manipulated by the function can be seen by visual inspection done by the developer. Source Code Analysis (SCA) is hence needed for developers, especially for large projects. In this work, the following tools are being examined which are made in respect to software visualization:-

1. Kieker [1]
2. Octobubbles [2]
3. Umple [3]
4. Reprograms [4]
5. ClonEvol [5]
6. CVSscan [6]
7. Revision Tower [7]
8. 3-DSoftVis [16]
9. History Slicing [8]
10. RelVis [9]
11. Chronia [10]
12. Spectographs [11]
13. Evolution radar [12]
14. CodeCity [13]

A source code is created by the developer or a programmer. However, the maintenance engineer has equal stake in the source code. The maintenance engineer needs to undertake the changes to the software. Thus in addition the effect of changes also needs to be studied. If any support in the form of a visualization tool can be provided, then it is helpful to both developer as well the maintenance engineer.

II BACKGROUND

Mapping of the source code is done by making use of different tools developed for the source code. The mapping between the source code can be done either intra-document or inter-document.

Mapping and visualization in the Software Development Life Cycle have been continuous and evolving due to various issues. Analysis artifacts are SRS, design phase has SDS, and coding phase has source code. These artifacts have natural language text or semi-natural language text as the medium of communication. Keyphrase Extraction can be used to extract relevant features from different API documents and the source code. KE paves way for interpretation and understanding using NLP models. There are various alternative statistical and mathematical models which help in interpretation. The analysis of API documentation and the source code can hence help in developing traceability links and provide an advantage to the developers, analysts, and maintenance engineer. The visualizations can help analyst, developer and maintenance engineer in the following ways:

1. Advantages to the developer include:
- Ability to visualize the changes done on source code.
- Ability to check relevant linkage between the design and source code developed.
- Visual inspection of large software projects helps in better understanding.

2. Advantages to the Software Maintenance: Software maintainers are tasked with introducing changeability through Software Configuration Management (SCM). This change is done by studying the end user's requirements. Hence, software maintenance work involves the knowledge of analysis, design, coding, and testing as well. A visual inspection of source code can aid in a quick understanding of how software changes can cause effects in different phases of the Software Development Life Cycle (SDLC).

In the analysis phase of agile development, these visualizations are used in XP, SCRUM methodologies for gaining better understandability of requirements. In the design phase, the visualization and its corresponding

mappings are done by UML diagrams. UML diagrams aid in providing a vision of software to be developed by the developer. UML diagram address both the problem domain as well as the solution domain of a project. Hence, its visual representation is essential for giving an idea about the software to be developed in the context of large projects. In the coding phase of SDLC, there is a technique of Source Code Analysis (SCA) which is done on the source code. The developers get benefitted from such analysis techniques as better clarity on the program flow and data flow is obtained.

In the field of software testing, there are various techniques such as the boundary value analysis (BVA) and unit path testing which allows visualization of source code. There are many tools both automated and semi-automated tools which focus on the development of different skill sets. Source Code Analysis (SCA) consists of model construction, pattern knowledge, analysis, and pattern recognition [15]. Model construction involves creating Abstract Syntax Tree (AST), Concrete Syntax Tree (CST) from the source code. The AST hence created forms the basis for the next step in SCA. AST pattern knowledge and recognition try to understand the pattern behind the source code. Figure-1 shows the basic steps in SCA. This paper hence tries to discuss such existing tools that are available for the visualization and mapping of software.

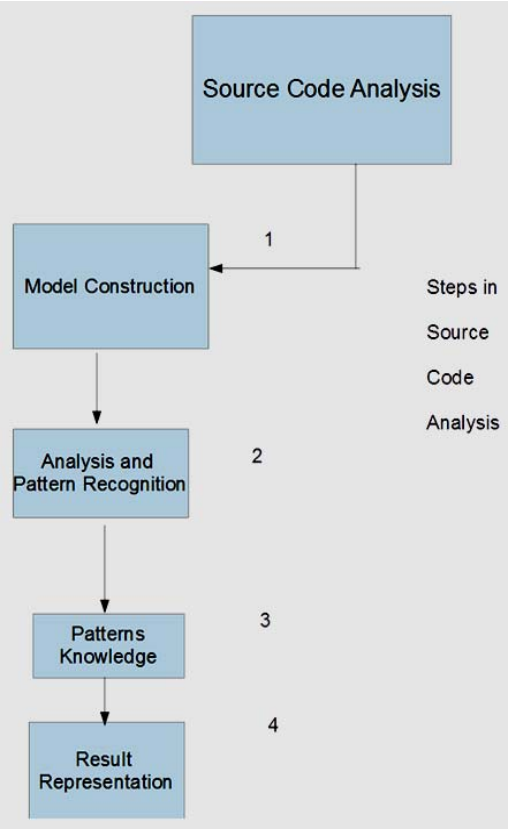

Figure 1. Source Code Analysis Steps [15]

III. LITERATURE REVIEW

Table-I shows some of the tools developed, source code of the software visualization tool, research methodologies used in these tools.

Specific to UML Models, Octobubbles provides a multidimensional view of the source code in a visual environment. Octobubbles intends to show the parallel visualization and synchronization of software models [2].

A. C/C++/Java Related Visualization Tool

ClonEval is a software tool which takes as input source code written in C, C++ and Java programming language. This tool undertakes evaluation of the input source code on the parameters of project, file and scope. The visualization includes making use of a mirrored tree for displaying the file and its scope structures while the edges represent the cluster relationships. The Doxygen and Simian are used in the background for accomplishing the tasks of static analyzer and code detector [3].

Kieker tool is developed for a web based GUI environment. The programming language used in developing Kieker is java programming language. The source code language acceptable for input is Java, .Net and Cobol. Kieker allows for checking the runtime nature of the system developed in Java,.Net and Cobol [1].

3DSoftVis makes use of different technologies such as Java, JDBC, RMI. The aim of 3DSoftVis is to provide visualization across software as it get changed across timelines [16].

B. General Visualization Tools

Chronos is a software visualization tool which allows querying, exploration and discovery of historical changes in source code. The tool is said to be better than 'diff' as it does provide all historical versions. The tool provides a zoom-able interface which allows querying both high level views as well as low level views [4].

CVScan makes use of a version centric approach to software visualization, representation and evolution. The tool provides a line based approach in seeing the changes across different versions of code. The tools usage is checked using data sets from real-world [6].

Revision Towers is a tool which makes use of log files in visualizing version control. The two log files are viewed in parallel. The central section shows software release within a log file. Towers are displayed in a grid. Revision towers finds similarities with 3DSoftVis software [7].

Code City is a language-independent tool for analyzing large software codes. The basis of this software is the on the premise that classes are buildings and packages are districts. The tool is applied to large scale projects or evaluation purposes [13].

Chronos is a tool implemented for creating history slicing. Chronos allows for checking the entire code across different versions which are of interest to the developer. The experimental results show that Chronos provides better results as compared to other techniques [8].

Umple accepts as input a generalized programming language. Umple is itself a programming language which allows easy integration of implementation with the design artifacts such as UML diagrams [3].

Reprograms tool is a metric-based visualization tool. This tool accepts as input any repository url. An instance of such repository is Github.

Evolution metrics make use of property based measurement in visualization of software artifacts [11].

Evolution Radar accepts as input Mozilla's source code as a repository. The tool allows representation of coupling both as document and at logical level coupling [12].

TABLE 1. SUMMARY OF TOOLS DEVELOPED FOR DIFFERENT LANGUAGES FOR SOFTWARE VISUALIZATION

Reference No.	Source Code Language	Name of Tool Developed	Research Tools used	Work Done	Year of Publication
1	Java,.Net, Cobol	Kieker	Java Programming Language, Web based GUI	Provides a framework which helps to oversee and analyze the runtime behavior of system. Kieker is currently deployed in multiple domains of Software Engineering. [18].	2012
2	UML Models	Octobubbles	Not Mentioned	Provides a multi-dimensional view interactive environment for parallel visualization and synchronization of software models	2018
5	C, C++, Java	ClonEvol	C,C++,Java	ClonEval takes information from got from software versioning	2013
15	Matlab/Octave	None Mentioned	Knowledge Discovery Meta-model	The authors generate KDM instances which are fed as input for further processing and analysis.	2018
3	General purpose programming language for visualization	Umple	Programming language	Umple is a language which integrates modeling and implementation.	2016
4	Repository based projects in any programming language	Reprograms	Metric based visualization model	Tool for checking the differences and similarity of software projects.	2016
16	Not mentioned	3DSoftVis	Java, JavaScript, webserver,VRMLJDBC, RMI	Tool for visualizing the software's release history as it get changed across timelines	2000
9	Not mentioned	RelVis	Kiviat Graphs	The tool allows studying the summary of released history of the source codes. This aids in visual inspection across different commits.	2005
6	Not Mentioned	CVSscan	Not mentioned	Line based approach for changes across different versions	2005
7	Not Mentioned	Revision Towers	Not Mentioned	Creates a visualization across different versions of software's	2002
8	Not Mentioned	Code City	Small Talk, Eclipse IDE	Code city is Eclipse based plugin which makes use of small talk	2011
9	Not Mentioned	Chronos	Online tool	Develops a tool named chronos for showing implementation	2012
11	Not Mentioned	Evolution Spectrographs	Not mentioned	a visualization mechanism using property based measurements for the components of source code	2004
12	Mozilla repository source code	Evolution Radar	Not mentioned	Allows visual representation of the given code at both document level as well as module level logical coupling. The coupling at various levels are shown by the tool	2006

IV. SNAPSHOT OF SOME SCA TOOLS

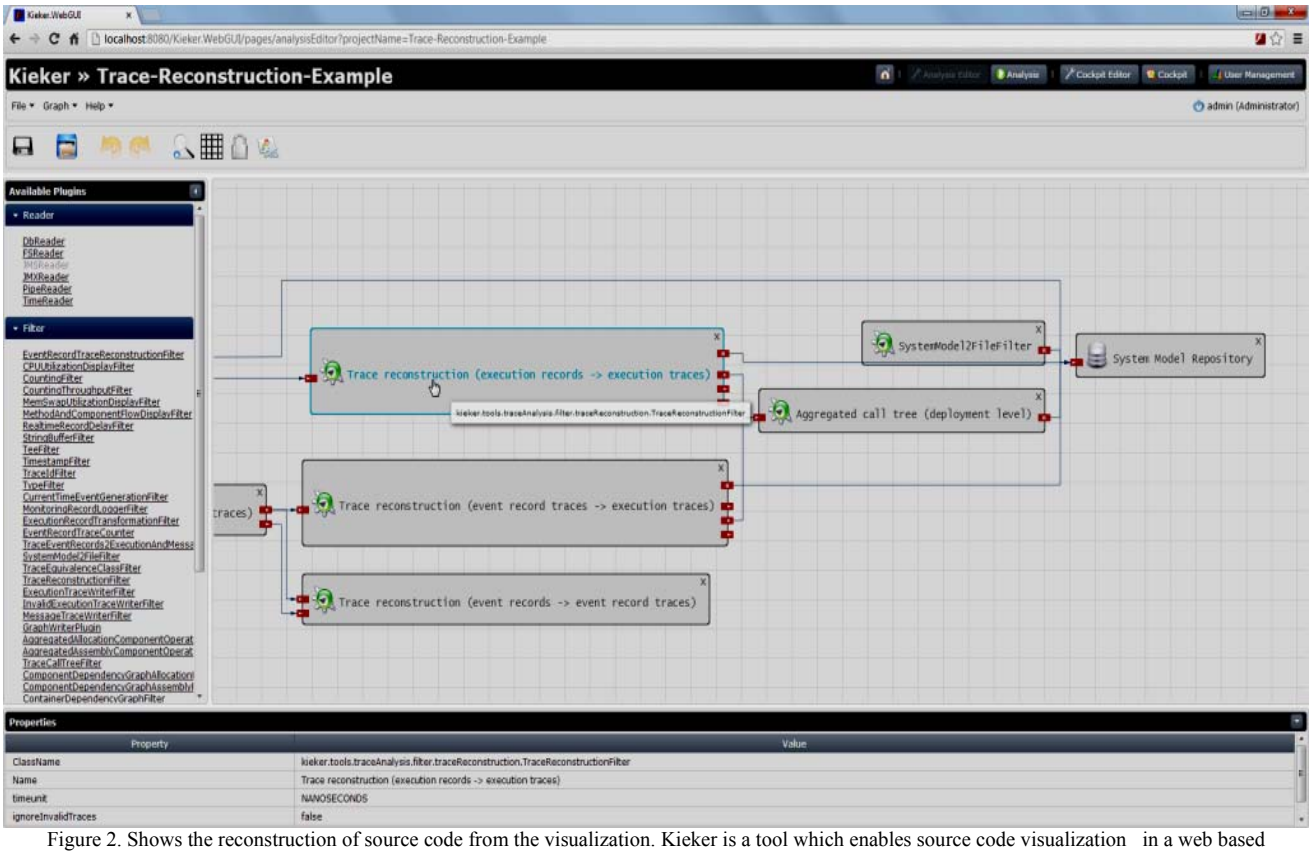

Figure 2. Shows the reconstruction of source code from the visualization. Kieker is a tool which enables source code visualization in a web based environment.

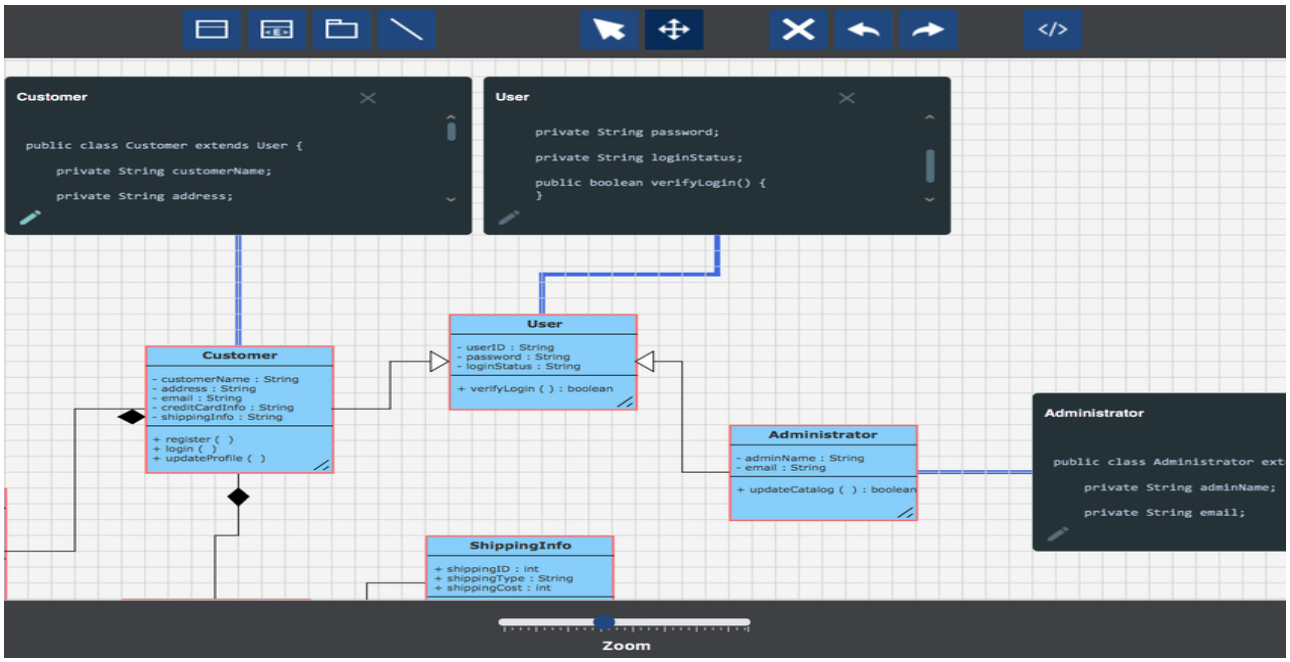

Figure 3. Source code and Its corresponding UML class diagram as shown in the tool

Octobubbles allow direct mapping of source code to its visualization representation in the form of UML diagrams. Figure-3 shows how in Octobubbles the source code is connected with its diagrammatic representation of UML representation.

V. OBSERVATION FROM LITERATURE REVIEW

There were the following findings from the literature review undertaken:

1. In a cloud-based environment, there were no specific visualization tools developed focusing on larger project source-files.
2. Many object-oriented programming languages such as small talk do not have any source code visualization tool.
3. In the source code visualization's developed, the architecture of the system is not recovered.
4. The solution domain is only addressed in the source code visualizations which is not merged with the problem domain analysis.
5. The source code visualization's developed were either semi-automated completely automated.

The ultimate aim of any SCA tool is to develop visualization or summarization of the given source code. The source code must hence be passed through the processes of model construction, analysis and pattern recognition, pattern knowledge, result interpretation.

Model construction creates various graphs as intermediate entities. These intermediate entities are fed for pattern recognition, pattern knowledge, and result interpretation.

TABLE II. GIVES THE ADVANTAGES AND DISADVANTAGES OF SOFTWARE VISUALIZATION TOOLS DEVELOPED

Reference No.	Source Code Language	Name of Tool Developed	Advantages of The Tool	Disadvantage of The Tool
1, 18	Java,.Net, Cobol	Kieker	Allows the user to monitor the run-time behavior of distributed systems.	1, 18
2	UML Models	Octobubbles	Solves the problem of program comprehension and facilitates software navigation	2
5	C, C++, Java	ClonEvol	ClonEvol provides visual representation of software across different versions.	The scope-oriented view could provide better representation
6	Not Mentioned	CVSscan	Provides a visual tool for inspecting contributions of developers, statement by statement.	Not Mentioned
7	Not Mentioned	Revision Towers	Provides an interface for the user to see the current working areas of the project, the change frequency and how work are shared in a project.	The tool did not have any link with the version-control systems.
8	Not Mentioned	Code City	Code City facilitates reduced task completion time and correctness of the task performed.	Code city did not perform better in comparison to Eclipse and Excel tools combined together.
9	Not Mentioned	Chronos	The developer gets benefited as deelopers had study reduced information to get to results. The accuracy of the developers results also increased for software maintenance.	None Mentioned
3	General purpose programming language for visualization	Umple	1. The tool is more useful to developers due to its textual coding ability	Umple is restricted to English natural language
4	Repository based projects in any programming language	Reprograms	Tool provides user with interface to filter out software projects and make more rational decisions for evaluations.	None Mentioned
16	Not mentioned	3DSoftVis	Tool for visual representation of release history of the project.	None mentioned
9	Not mentioned	RelVis	Tool provides visualization of of graphical representation of source code extending up to n releases.	3D Kivit diagrams are not drawn and are part of future scope.
11	Not Mentioned	Evolution Spectrographs	The tool highlights the changes across software releases.	None Mentioned
12	Mozilla repository source code	Evolution Radar	Tools facilitates visual representation at document and module level. The tool also does analysis at different level of logical coupling between objects at varying granularity.	Integration with different web based tools.

TABLE III. ARTIFACTS RELATED TO SOURCE-CODE OF A PROJECT

Sr. No.	Type of Document	Artifact of SDLC Phase	Usage	Usage In SCA Analysis
1	Dynamic Link Library (DLL)	Coding	Use in proving function calls	Yes
2	Software Requirement Specification (SRS)	Analysis	Useful in providing requirement as official document between clients	Yes
3	Software Design Specification (SDS)	Design	Useful in providing Design level specification of the software being developed	Yes
4	Log Files	General	Create log of any activity	Yes*
5	User Manual	Deployment	Helps in user's day-to-day usage of software being developed	No
6	Meeting Records	General	For creating proof of points discussed in meeting	Yes*
7	Temp files	General	For storing temporary information	Yes*
* depends on the relation with the source code being fed as input				

VI. RECOMMENDATIONS ON THE LITERATURE REVIEW

The literature survey is full of Source Code Analysis tools. However, the author suggests the following research directions for creating tools that are addressing a wider audience and phases of SDLC. Authors propose that NLP and AI technologies (NLP and ML) can aid in better software visualization. Source Code consists of text in-form of the developer's comments in the source-code files in addition to the programming language statements.

The programming language statements can be understood by creating a parallel lookup dictionary or be learnt to a machine using ML/DL techniques.

The existing tools in SCA focus on mainly the source code as an input. The developers of such tool need to adopt a more holistic approach while developing these SCA tools. This is because source code ultimately has not just the programming files but also various allied files which make up the project.

VII. CONCLUSION

This is a literature review regarding existing software visualization tools developed in SCA. The literature review is done on software visualization and its related fields. The literature contains various tools for providing visualization to the software developer and maintenance engineers. There has been more work focused on software visualization with more traditional languages such as C, C++, java. The current paper also provides the future direction wrt other software artifacts onto which similar visualization can be conducted/ The software visualization wrt other software tools is not available. The future work includes conducting a Systematic Literature Review (SLR) in order to provide more exhaustive study SCA tools.

REFERENCES

[1] Van Hoorn, André, Waller, Jan and Hasselbring, Wilhelm, "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis" [Paper] In: 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE'12), April 22-25, 2012, Boston, Massachusetts, USA, 2012.

[2] Rodi Jolak, Khan-Duy Le, Kaan Burak Sener, Michel R.V. Chaudron, "OctoBubbles: A Multi-view Interactive Environment for Concurrent Visualization and Synchronization of UML Models and Code", SANER 2018, Campobasso, Italy, ERA Track, page 482-486, 978-1-5386-4969-5 , 2018 IEEE.

[3] T. C. Lethbridge, V. Abdelzad, M. H. Orabi, A. H. Orabi, and O. Adesina. "Merging modeling and programming using Umple". In International Symposium on Leveraging Applications of Formal Methods, pages 187–197. Springer, 2016.

[4] D. Rozenberg, I. Beschastnikh, F. Kosmale, V. Poser, H. Becker, M. Palyart, G. C. Murphy, "Comparing repositories visually with reprograms", in: Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, ACM, New York, NY, USA, 2016, pp. 109–120. doi:10.1145/2901739.2901768.

[5] A. Hanjali´c, "Clonevol: Visualizing software evolution with code clones", in: 2013 First IEEE Working Conference on Software Visualization (VISSOFT), 2013, pp. 1–4. doi:10.1109/VISSOFT.2013.6650525.

[6] L. Voinea, A. Telea, J. J. van Wijk, "CVSscan: visualization of code evolution", in: Proceedings of 2005 ACM Symposium on Software Visualization (Softviz 2005), St. Louis, Missouri, USA, 2005, pp. 47–56.

[7] C. Taylor, M. Munro, "Revision towers", in: Proceedings 1st International Workshop on Visualizing Software for Understanding and Analysis, IEEE Computer Society, Los Alamitos CA, 2002, pp. 43–50.

[8] F. Servant, J. A. Jones, "History slicing: Assisting code-evolution tasks", in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE '12, ACM, New York, NY, USA, 2012, pp. 43:1–43:11. doi:10.1145/2393596.2393646 URL http://doi.acm.org/10.1145/2393596.2393646

[9] M. Pinzger, H. Gall, M. Fischer, M. Lanza, "Visualizing multiple evolution metrics", in: Proceedings of SoftVis 2005 (2nd ACM Symposium on Software Visualization), St. Louis, Missouri, USA, 2005, pp. 67–75.

[10] T. Gˆırba, A. Kuhn, M. Seeberger, S. Ducasse, "How developers drive software evolution", in: Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2005), IEEE Computer Society Press, 2005, pp. 113–122. doi:10.1109/IWPSE.2005.21. URL http://scg.unibe.ch/archive/papers/Girb05cOwnershipMap.pdf

[11] J. Wu, R. Holt, A. Hassan, "Exploring software evolution using spectrographs", in: Proceedings of 11th Working Conference on Reverse Engineering (WCRE 2004), IEEE Computer Society Press, Los Alamitos CA, 2004, pp. 80–89.

[12] M. D'Ambros, M. Lanza, M. Lungu, "The evolution radar: Integrating fine-grained and coarse-grained logical coupling information", in: Proceedings of MSR 2006 (3rd International Workshop on Mining Software Repositories), 2006, pp. 26 – 32.

[13] R. Wettel, M. Lanza, R. Robbes, "Software systems as cities: a controlled experiment", in: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, ACM, New York, NY, USA, 2011, pp. 551–560. doi:10.1145/1985793.1985868.

[14] Thiago de Lima Mariano, Glauco de Figueiredo Carneiro, Miguel Pessoa Monteiro, Fernando Britoe Abreu and Ethan Munson, "A

Parser and a Software Visualization Environment to Support the Comprehension of MATLAB/Octave Programs", In Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018) - Volume 2, pages 179-186, ISBN: 978-989-758-298-1.

[15] Radoslav Kirkov, Gennady Agre, "Source Code Analysis – An Overview", Cybernetics And Information Technologies, Volume 10, No 2, Bulgarian Academy Of Sciences, 2010.

[16] Claudio Riva, "Visualizing Software Release Histories with 3DSoftVis": Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium,Limerick, Ireland. DOI: 10.1145/337180.337644 ,ISBN: 1-58113-206-9.

[17] Matthias Junker, "Kumpel : Visual Exploration of File Histories", Masters Thesis, University of Institut f˙ur Informatik und angewandte Mathematik, 2008.

[18] Hasselbring, W., & van Hoorn, A. (2020). Kieker: A monitoring framework for software engineering research. Software Impacts, 100019.

AUTHORS' PROFILE

Nakul Sharma completed the B.Tech in I.T. from Bharati Vidyapeeth, Pune. He did a Master of Engineering in Software Engineering from Thapar University in the year 2011. He has taught several subjects at PG and UG level. Currently he is pursuing PhD from K L University. His research interests include text mining, natural language processing and software engineering.

Prasanth Yalla received his B.Tech Degree from Acharya Nagarjuna University, Guntur (Dist), India in 2001, M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University in 2004, and received his Ph.D. degree in CSE titled "A Generic Framework to identify and execute functional test cases for services based on Web Service Description Language" from Acharya Nagarjuna University, Guntur (Dist), India in April 2013. He was an associate professor, with Department of Information Science and Technology in KL University, from 2004 to 2010. Later he worked as Associate professor, with the department of Freshman Engineering from 2011 in KL University. Presently he is working as Professor in the department of Computer Science & Engineering in KL University and also Associate Dean (R&D) looking after the faculty publications. Till now he has published 28 papers in various international journals and 4 papers in conferences. His research interests include Software Engineering, Web services and SOA. He taught several subjects like Multimedia technologies, Distributed Systems, Advanced Software Engineering, Object Oriented Analysis and design, C programming, Object-Oriented programming with C++, Operating Systems , Database management systems, UML etc. He is the Life member of CSI and received "Active Participation- Young Member" Award on 13-12-13 from CSI. He has applied a project to SERB very recently.