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. Umlple [3]
6. CVSscan [6]
7. Revision Tower [7]
8. 3-DSoftVis [16]
9. History Slicing [8]
10. RelVis [9]
11. Chronia [10]
13. Evolution radar [12]

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, SCUM 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.

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

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


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.