Source Code Explorer System to Enhance the Software Maintainability and Scalability

Mohammad Subhi Al-Batah, Nouh Alhindawi, Saleh Oqeili, and Obaida M. Al-Hazaimeh

***Abstract*—**Typically, the traditional ways of exploring any artifacts relate to the source code through software systems is useless and considered as error prone and time consuming. Software systems are often comprised of many lines of code scattered across many different files, all located within a complex hierarchical file system. This hierarchical distribution of the code fragments makes the process of changing the source code complex task and need to be done from an expert developer to avoid affecting or changing unrelated fragments. Due to this complexity, developers can easily become disoriented and lost within software systems, moreover the developers cannot also extract data about the code. Because of these problems, in this paper, a tool called Source Code Explorer (SCodeEx) is presented. The proposed tool has been developed using C#. SCodeEx can solve the problem of the use of paper-based solutions. SCodeEx is solving the problem of categorizing and extracting data from the code in professional way. The proposed system is successfully passed and the users notice the credibility of the system to explore the classes with its functions and names, number of callee, call, variables, return data type and access modifier to each function. Compared to existing tools, SCodeEx is more professional and the way to solve the problem is faster and easier than others.

***Keywords*—**Software testing, Software development, Quality system, Software complexity, Code visualization, Code navigation

I. INTRODUCTION

IN the last few decades, a lot of researches has been directed to enhance the programming skills, this is due to the fact that the style of source code implementation affect the whole life cycle of the application and therefore it effect the time of developers and users [1]. Moreover, writing and implementing any system in a professional way will positively appear on the usage of the program as well as on the process of evolution and maintenance of the undertaken system. There are noticeable disagreements in the environment of software development that may arise between the developers and their team leaders or between developers and testers [2]. The accusation is that the developer who created code is not efficient, the question that arises: "How the evaluation process was conducted without relying on a fixed mechanism?" Since there is no reference to the rule of permission may potentially this provision not right, or rather would not be fair, even for the tester can be performed to give information about the functions in the code that it is efficient and is not so, and when it is delivered to the organization that bought the system will discover that it really inefficient, and this damaging large [3].

There are a lot of tools that were built in order to help in program understanding, and to simplify the comprehension task for a maintainer [4]. For instance, SNiFF is one of the best well-known commercial tools, and it was produced to assist in source code understanding and to facilitate maintenance tasks. Ghinsu [5] is a program understanding framework described, and SeeSoft [6] is a tool for visualizing software statistics from a variety of sources. Such tools are helping drastically in improving and accelerating a developer's overview of complex system software [7]. Moreover, those tools have practical benefits in terms of generating fewer bugs or an easier time comprehending a new piece of source code.

In addition, researchers with the goal of improving the comprehension process and saving developer's time and effort have presented a set of recommended tools to guide system software navigation while exploring and understanding a system. Mylar [8] used a degree-of-interest model to distinguish and mark the non-relevant files from the file explorer in Eclipse. NavTracks [9] supported a tool that recommends which files are related to the currently chosen files. Deline et al. [10] presented a framework to improve the software navigation process. On the other hand, Robillard [11], presented a FEAT tool that is capable of providing suggestions using graphs manually created by users, to enhance navigation effectiveness and improve the comprehension process. RedHat Source-Navigator is another tool that is being developed to assist in understanding the complex system software.

M. S. Al-Batah is with the Faculty of Sciences and Information Technology, Jadara University, Irbid, Jordan (e-mail: albatah@jadara.edu.jo)

N. Alhindawi is with the Faculty of Sciences and Information Technology, Jadara University, Irbid, Jordan (e-mail: hindawi@jadara.edu.jo)

S. Oqeili is with the Computer Engineering Department, Jadara University, Jordan, on leave from the Al Balqa Applied University, Jordan (e-mail: saleh@bau.edu.jo)

O. M. Al-Hazaimeh is with Computer Science & Information Technology Departments at Al Balqa Applied University, Jordan (e-mail: dr_obaida@bau.edu.jo

The Searchable Bookshelf [12] designed to help in producing and navigating software structure diagrams. It enables the users to visualize different aspects of a software system (sub-system, files…etc.) using diagrams shapes, and it also shows interactions between the different system components. SHriMP [13], employs the hyperlinks in order to navigate the source code, and gives a better view of the source code components.

In software engineering, many Researchers suggested and used alternative approaches that do not involve giving great amounts of attention to software comprehension. Examples of such approaches include refactoring. Refactoring [14] tries to improve the software's interior construction, maintainability, and comprehensibility, without changing software's behavior/functionality.

Moreover, the researchers have developed a lot of tools to help in code comprehension, these tools stand mainly on extracting function calls from source code [15]. For example, Brilliant source code browser, it can import sources in many different languages, and split them down into classes/methods/functions, Exploration Tools; it is a command-line based set of tools for examining functions and the structure of C source code, it allows the user to scan and analyze source code to build function call hierarchy and data structure relations, and Source Navigator tool; it is known as source code comprehension and documentation tool, it allows the developers to perform source browsing, showing relationships (call/callby/include/includeby/etc.) between the various parts of the program [16].

In [17], the authors presented call-extraction tool, namely callextractor, their tool can perform ordered-pattern extraction. In [18], the authors used the function calls for source code directed testing of functional programs. The authors use call graphs in the context of software measurement for functional programs. They consider function calls as atomic operations and are produced for each function independently. In [19], a code search system known as Portfolio is introduced. Portfolio is a tool supports and helps programmers in identifying the relevant functions or fragments of source code that implement a specific concept that are reflected in developer query expression, and determining how these functions are well relevant to the query, moreover, the tool also make visualizing dependencies of the retrieved functions to show their flows. In [20], the authors use function calls as a guide in order to do local and global analysis in source code by finding paths in the control-flow graphs of functions. The author concluded that identifying the list of functions that called from a given function, can help in better understanding of source code specially for large and complex programs.

In this paper, we present SCodeEx as a tool that supports explorer through software systems. The program works with all types of computers that use the Windows operating system. It supports a range of services that require the developer first degree, by giving data on the functions that in the code, such as the degree of priority of each function, and the classes that called every function. We begin with a brief survey of current tools for source code explorer. Then, the proposed SCodeEx is presented. This is followed by a discussion of the design, implementation and testing of SCodeEx. Finally, conclusion, recommendations and future work are presented.

II. RELATED WORKS

There are different tools and mechanisms exist in state of art for analyzing and exploring the source code. But, based on literature review and a high citation rate, Ghinsu, SeeSoft, and NavTracks are the famous tools. These tools were built in order to help in program understanding, and to simplify the comprehension task for a maintainer.

A. *Ghinsu*

The Ghinsu project started in early 1991 at the University of Florida's Computer and Information Sciences Department and has been funded by the Software Engineering Research Center (SERC). Its target is the development of an environment that integrates a number of tools aiding in a number of software engineering activities, primarily in software maintenance. The current version of Ghinsu can handle multiple file programs, written in a large subset of ANSI-C, and has a graphical user interface. It can perform a variety of program analysis functions, including: program slicing and dicing, ripple analysis, calculation of reaching definitions, calculation of DU and UD chains, and calculation of execution slices.

B. *SeeSoft*

Seesoft software visualization system allows the developer to analyze up to 50000 lines of code simultaneously by mapping each line of code into a thin row. The color of each row indicates a statistic of interest, e.g., red rows are those most recently changed, and blue are those least recently changed. Seesoft displays data derived from a variety of sources, such as version control systems that track the age, programmer, and purpose of the code (e.g., control ISDN lamps, fix bug in call forwarding); static analyses, (e.g., locations where functions are called); and dynamic analyses (e.g., profiling). By means of direct manipulation and high interaction graphics, the user can manipulate this reduced representation of the code in order to find interesting patterns. Further insight is obtained by using additional windows to display the actual code. Potential applications for Seesoft

include discovery, project management, code tuning, and analysis of development methodologies.

C. *NavTracks*

NavTracks is a tool that supports navigating through software systems. NavTracks keeps track of the navigation history of software developers, forming associations between related files. These associations are then used as the basis for recommending potentially related files as a developer navigates the software system.

Despite of the large number of source exploring tools, still we need for more efficient tools that can support more accurate analysis and exploring for developers. So, better tool for source code explorer are required.

This paper proposes SCodeEx. The SCodeEx motivates heads of work, testers, and all decision makers in the company of the software to take accurate, well-structured, and well-presented information about the code. The main objective of SCodeEx can be summarized like follow:

1. To save time, money and reduce the needed effort; the costs that are software companies dispensed with the aim of tracking code may outweigh the costs of creating it from scratch, but in terms of time, huge amount of time are usually wasted by developers, whether developers, analyzers, testers or even leaders, lost the time while trying to keep track the code in terms of mental effort, the process of tracking of the most important requirements it's focus and understanding carefully and all these things are considered difficult for people or in some cases is almost impossible, so why not be a program does all this work.

2. Increasing the accuracy of retrieved data. The main contribution of the presented tool is basically providing the developers with information about particular function, variable, or class in the source code. Thus, the manual manipulation over any source regarding finding or recovering any related information for any fragment of code may has large percentage of error, moreover, the manual investigation may also waste the time of developers.

3. An accurate statistical report will be provided for each function; the extracted report has many different options which can be easily added to the report.

Other existing programs provide services to help software developers to determine the efficiency of the code, where it give just a comparison between the existing code and the standard one, but the SCodeEx will support additional services not in the short term but in the long term, in terms of targeting and maintain the system in the future easily and efficiently.

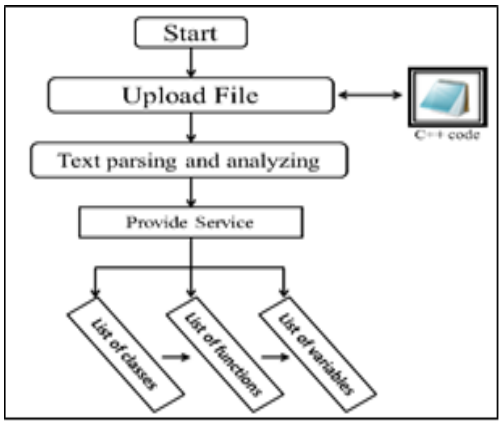


Fig. 1 Flow Chart for the main process

III. PROPOSED APPROACH

As mentioned before, the main objective of SCodeEx is to provide data about the code to be classified and analyzed later by the owner and developers of the source code easily and accurately. The SCodeEx presents the artifacts of source code in appropriate data structure to be read and retrieved easily. This new structure can help efficiently while the evolution and refactoring processes over any undertaken software. More details are explained later in this section.

Generally, the presented tool works in a sequence manner, the program input is a C++ code, then the program will perform some preprocessing steps over the code, and then the output will show hints to developer about the code for more details. The SCodeEx system is designed with simple and accurate user interface that can be directed and used by the developer effortlessly, even for the novice users. The flow chart for the SCodeEx can be illustrated by Fig. 1. The initial step in the work flow is uploading and selecting the source file, typically, the source file is a C++ program. By selecting the source file, the user can explore the artifacts of the undertaken system. This step is considered the base step for the whole process.

Many different outputs for the presented tool can be offered for the users such as providing the number of functions, class, and variables in the system, providing the names of classes and the functions in the system, providing name of all local variables in the system and its data type, providing caller and caller for each function along with numbers, finding the ratio of priority for each function, providing data about the types of functions, and providing access to modify the function directly.

The following sub-sections present briefly the needed steps for running the tool a long with a snip for each step and the reader can easily follow the steps in order to explore any system structure. The main features and buttons are also presented.

A. INSTALLATION

We test the .exe file of the program in installation process. The way of installing SCodeEx system on the computer is to copy the program icon to the desired device. The install process is not complex, and eases of dealing with users. The SCodeEx system contains the following screens; welcome, upload file, general table, graph tree, search, add and edit class, and main screen.

B. UPLOAD FILE SCREEN

After running the program, the welcome screen appears for the user also known as Sign-in Screen, the user then press on the "Enter" button, next, a dialog box will appear which ask the user to select the source file to be uploaded as shown in Fig. 2. The Browse button also added to the screen to select the source file, once the file is chosen and uploaded, the explore button will be enabled.

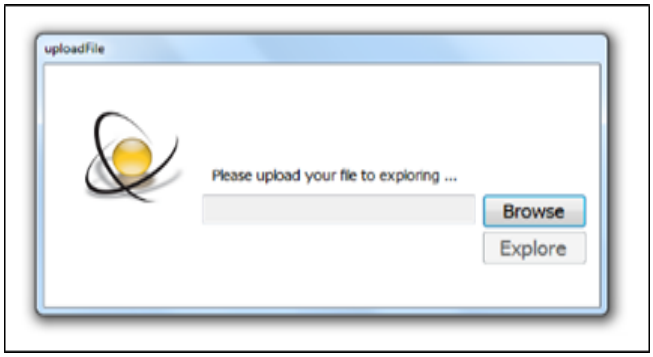


Fig. 2 Upload screen

C. GENERAL TABLE SCREEN

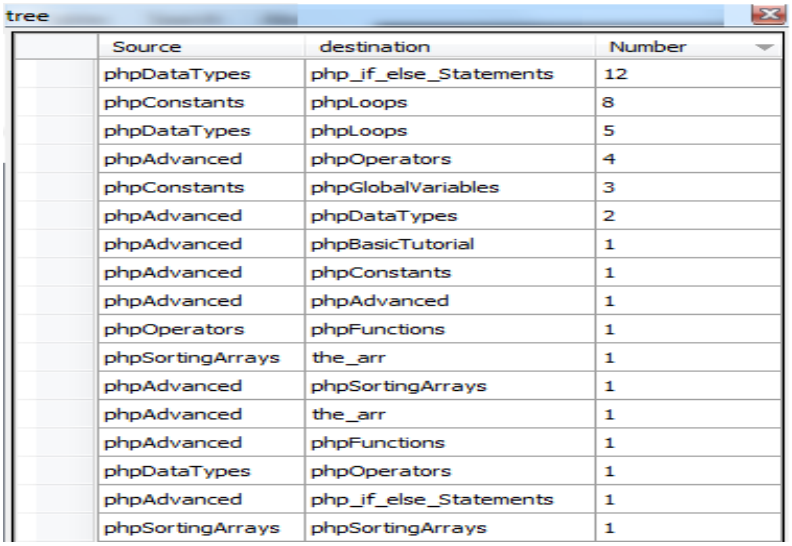
The general table shows a specific data or profile for each function in the system which include: class name, number of local variables, number of call and callee, access modifiers, and return data type as shown in Fig. 3.

D. GRAPH TREE TABLE SCREEN

The graph tree table shows the relations between classes by showing the number of calling between them as shown in Fig. 4. For example, the table provides number of calls between: phpDataTypes→phpLoops (15-times). So, the phpDataTypes class based on phpLoops in a high degree.

Function	Number Of local variables	Number Of callee	Number Of call	Class	Access modifiers	Return data type
IntroOfPHP	0	0	0	AboutTheSystem	private	void
phpVariables	1	1	2	phpAdvanced	private	void
phpVarScope	1	0	1	phpAdvanced	private	void
localGlobalScope	1	0	1	phpAdvanced	private	void
php_global_Keyword	1	0	3	phpAdvanced	private	void
php_static_Keyword	1	0	4	phpAdvanced	private	void
php_echo_Statement	1	0	2	phpAdvanced	private	void
phpStrings	0	1	5	phpDataTypes	private	void
phpIntegers	3	0	4	phpDataTypes	private	void
phpFloatingPointNum	1	0	4	phpDataTypes	private	void
phpBooleans	2	1	0	phpDataTypes	private	void

Fig. 3 General table


tree

Source	destination	Number
phpDataTypes	php_if_else_Statements	12
phpConstants	phpLoops	8
phpDataTypes	phpLoops	5
phpAdvanced	phpOperators	4
phpConstants	phpGlobalVariables	3
phpAdvanced	phpDataTypes	2
phpAdvanced	phpBasicTutorial	1
phpAdvanced	phpConstants	1
phpAdvanced	phpAdvanced	1
phpOperators	phpFunctions	1
phpSortingArrays	the_arr	1
phpAdvanced	phpSortingArrays	1
phpAdvanced	the_arr	1
phpAdvanced	phpFunctions	1
phpDataTypes	phpOperators	1
phpAdvanced	php_if_else_Statements	1
phpSortingArrays	phpSortingArrays	1

Fig. 4 Graph tree table

E. SEARCH SCREEN

The search screen contains two tabs such as functions and variables as described below:

1) Function tab search

To test the search about function: Just select name of function from the list and then, press on the "Go" button and then, you can know the call and callee to the function (brief description) as shown in Fig. 5.

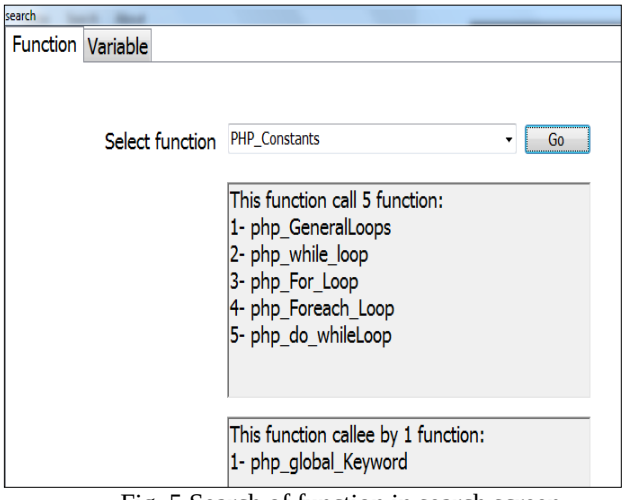


Fig. 5 Search of function in search screen

2) Variables tab search

To test the search about variables, we just select name of variable from the list and then press on the "Go" button and then you can explore the Scope (Class and function) to the undertaken variable as shown in Fig. 6.

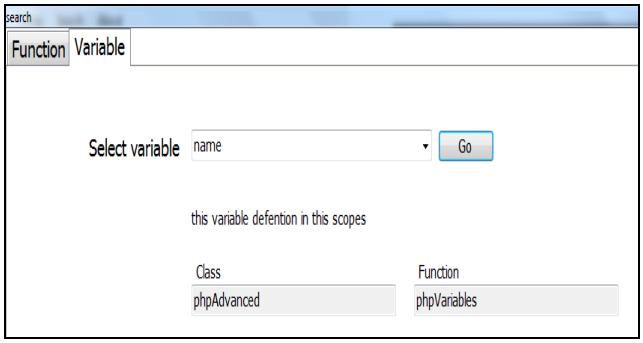


Fig. 6 Search of variable in search screen

F. ADD AND EDIT CLASS SCREEN

To add class on the selected code, the user need to press on the "Add Class" button, and then enter the class content as shown in Fig. 7.

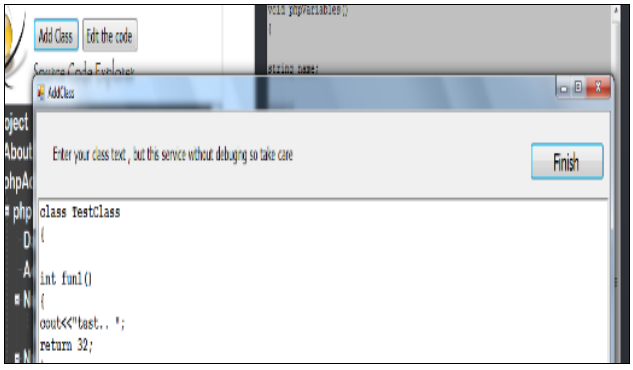


Fig. 7 How to add new class

Fig. 8 shows the menus for editing the added code. To edit the code, the user need to press on the "Edit the code" button and then edit from the source file as shown in Fig. 9.

```
//this code to the "Finish" Button :-
private void button1_Click(object sender, EventArgs e)
{
    bool ok = false;//this is flag
    MYproJ.Form1 ob = new MYproJ.Form1();//we want use method from From1
    code = ob.GetSourceCode() + "                                ";//the metod is : GetSourceCode()
    string newCode = "";//holder of updated code
    for (int i = 0; i < code.Length-10; i++)
    {
        //if find the word "class" then add the content of this class before it.
        if (ok==false && code[i] == 'c' && code[i + 1] == 'l' && code[i + 2] == 'a' && code[i + 3] == 's' && code[i + 4] == 's')
        {
            ok = true;
            newCode += "\n" + richTextBox1.Text + "\n";//add the code that written in the rich text box on the holder variable (newCode)
        }
            newCode += code[i];
    }

    StreamWriter sr = new StreamWriter(MyProj2.uploadFile.MyFile);//create channel to the file
    sr.Write(newCode);//write on the Source file
    sr.Flush();//flush the file
    sr.Close();//close the channel
}
```

Fig. 9 C# code that added the class to the source code

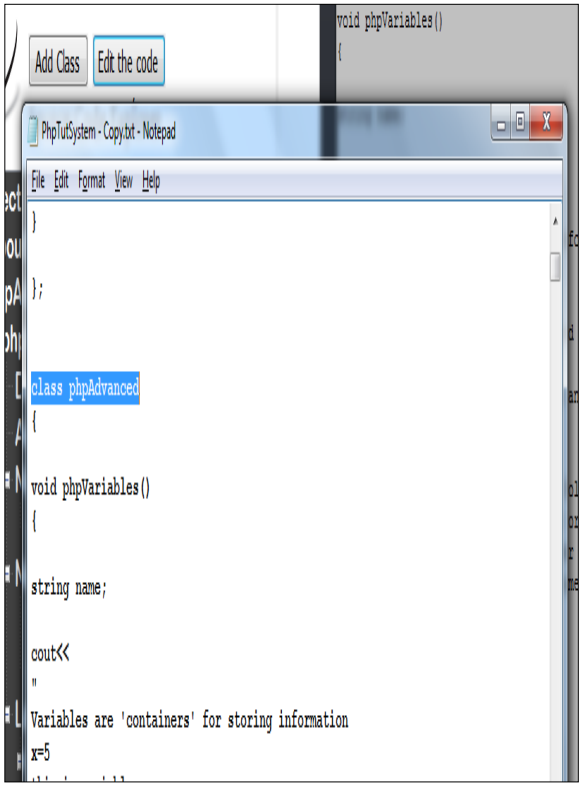


Fig. 8 How to edit the selected code

Fig. 10. Main screen

G. MAIN SOURCE CODE EXPLORER SCREEN

After uploading the file, the main screen will appear as in Fig. 10, which is divided into two parts: the right part is responsible for data display, and the left part is responsible for the selection (function or class).

The proposed SCodeEx system has been tested to a range of issues that we want to explore uploaded file (source code file). The proposed system is successfully passed and the users notice the credibility of the system in the following:

a) Explore the list of class.
b) Explore the function of a certain class.
c) Explore the return data type to each function.
d) Explore the access modifier to each function.
e) Explore the list of callee to each function.
f) Explore the list of call to each function.
g) Explore the list of local variables to each function.
h) Explore the code to each function.
i) Explore the code to each class.

As in example “explore the code to each class”. If pressed on the class name from the tree in the left part (See Figure 10), is assumed to be presented a code to this class on a screen which is located on the right. We have been tested these code to certain class, and it is actually the code to this class. If have been to take a snapshot (See Figure 11), see the content to class in original file and see the content on screen that provided by the SCodeEx system, you will notice the credibility of the system.

From the conducted evaluation and experiments, we can conclude that the proposed SCodeEx has many features such as exploring software code artifacts starting from class and ending by variables, and giving a statistical data about the source code. Another important feature is the visualization for the code based on class granularity, the visualization shows the relations between code fragments, in addition it show the dependency between code fragment. On other side, SCodeEX allows the developer to edit the code directly. Thus, the developers can directly see the impact of any added code, furthermore, this feature will make the coding more safely and more accurate.

IV. CONCLUSION

The result of this work is a practical system that includes a tool to help the developers, testers, team leaders and system analyzers to ease understand the large source code to find-out class, functions, and collection of data which relate to a particular system. This data can provide information about priority to each function and class. The presented tool enable the developers in viewing and exploring millions line of code easily and quickly. This is named SCodeEx. It can save time and money for Software Development Company by knowing the information about the code, and the report about fragments of code. Also the users can edit the code in easy way. As a future work, we plan to make a version of the presented tool to support Java language.

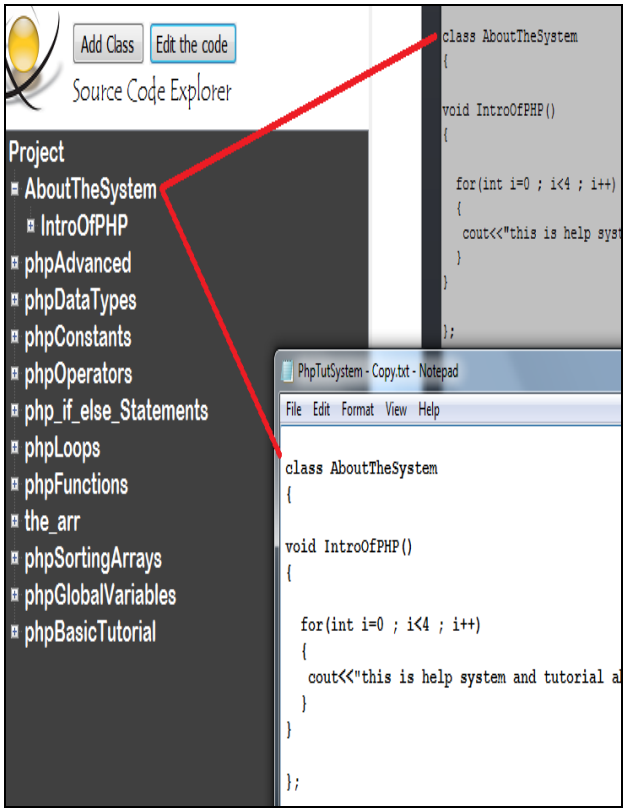


Fig. 11 Snapshot to test the credibility about the content to certain class

REFERENCES

[1] N. Alhindawi, R. Malkawi, M. S. Al-Batah, and A. Al-Zuraiqi, "Hybrid Technique for Java Code Complexity Analysis," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 8, pp. 379-385, 2017.

[2] O. Meqdadi, N. Alhindawi, M. L. Collard, and J. I. Maletic, "Towards understanding large-scale adaptive changes from version histories," in *2013 IEEE International Conference on Software Maintenance*, 2013, pp. 416-419.

[3] N. Alhindawi, J. Alsakran, A. Rodan, and H. Faris, "A survey of concepts location enhancement for program comprehension and maintenance," *Journal of Software Engineering and Applications*, vol. 7, p. 413, 2014.

[4] M. F. Klaib, M. S. Al-batah, and R. J. Rasras, "3-way Interaction Testing using the Tree Strategy," *Procedia Computer Science*, vol. 65, pp. 845-852, 2015.

[5] P. E. Livadas and S. D. Alden, "A toolset for program understanding," in *[1993] IEEE Second Workshop on Program Comprehension*, 1993, pp. 110-118.

[6] S. Eick, J. L. Steffen, and E. E. Sumner, "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Transactions on Software Engineering*, vol. 18, pp. 957-968, 1992.

[7] F. Niessink and H. Van Vliet, "Software maintenance from a service perspective," *Journal of Software Maintenance: Research and Practice*, vol. 12, pp. 103-120, 2000.

[8] M. Kersten and G. C. Murphy, "Mylar: a degree-of-interest model for IDEs," in *Proceedings of the 4th international conference on Aspect-oriented software development*, 2005, pp. 159-168.

[9] J. Singer, R. Elves, and M.-A. Storey, "NavTracks: Supporting Navigation in Software Maintenance " *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pp. 325-334, 2005

[10] R. DeLine, A. Khella, M. Czerwinski, and G. Robertson, "Towards understanding programs through wear-based filtering," *Proceedings of the 2005 ACM symposium on Software visualization*, pp. 183-192, 2005.

[11] M. P. Robillar and G. C. Murphy, "A Tool for Locating, Describing, and Analyzing Concerns in Source Code," *25th International Conference on Software Engineering*, pp. 822-823, 2003.

[12] S. E. Sim, C. L. A. Clarke, R. C. Holt, and A. M. Cox, "Browsing and Searching Software Architectures," *Proceedings of the International Conference on Software Maintenance*, pp. 381-390, 1999.

[13] M.-A. D. Storey and H. A. Miiller, "Manipulating and documenting software structures using SHriMP views," *Proceedings of International Conference on Software Maintenance*, pp. 275-284, 1995.

[14] M. Fowler, "Refactoring: improving the design of existing code," *Addison-Wesley Professional*, 1999.

[15] Y. Padioleau, L. Tan, and Y. Zhou, " Listening to programmers Taxonomies and characteristics of comments in operating system code," *Proceedings of the 31st International Conference on Software Engineering.* , pp. 331-341.

[16] N. Alhindawi, O. M. Al-Hazaimeh, R. Malkawi, and J. Alsakran, "A Topic Modeling Based Solution for Confirming Software Documentation Quality," *International Journal of Advanced Computer Science and Applications*, vol. 7, pp. 200-206, 2016.

[17] J. W. LASKI and B. KOREL, "A data flow oriented program testing strategy," *IEEE Transactions on Software Engineering*, pp. 347-354, 1983.

[18] K. v. d. Berg, "Software Mesurement and Functional Programming " *University of Twente*, 1995.

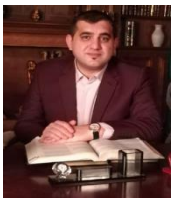
[19] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, "Portfolio: finding relevant functions and their usage," *Proceedings of the 33rd International Conference on Software Engineering*, pp. 111-120, 2011.

[20] G. J. Holzmann, "Static source code checking for user-defined properties," *Integrated Design and Process Technology (IDPT) Proc. IDPT*, vol. 2, 2002.

Mohammad Subhi Al-Batah received his PhD in Computer Science/ Artificial Intelligence from the University of Science Malaysia in 2009. After working as an assistant professor (from 2009) in the Dept. of Computer Science, Jadara Univ. in Jordan, he has been an associate professor at Jadara Univ. since 2014. He worked as a dean of Faculty of Science and Information Technology from 2015-2018. In 2019, he is the director of the Academic Development and Quality Assurance Center. His research interests include Image Processing, Artificial Intelligence, Medical Analysis, Real Time Classification and Software Engineering, E-mail: albatah@jadara.edu.jo, dralbatah@gmail.com.

Nouh Alhindawi is Associate Professor in Computer Science & Software Engineering Departments at Jadara University at Jordan since 2013. He is a member of the Software Engineering Development Laboratory (SDML). He is a member of the Software Traceability Development Laboratory (TraceLab). His research interests are in software maintenance and comprehension, using information retrieval in software engineering, and data visualization.

Saleh Oqeili is a full Professor of Computer engineering. He received his PhD in 1983. He worked in several Jordanian Universities and held different positions: head of computer science department, dean of IT faculties, and vice-president. Currently he is a staff member at Al Balqa Applied University and president at Jadara University/Jordan. Prof. Oqeili published more than 80 scientific papers and books in the areas of Computer Architecture, Computer Algorithms, Operating Systems, Reliability, Coding Theory, and Programming Languages.



Obaida M. Al-Hazaimeh is Associate Professor in Computer Science & Information Technology Departments at Al Balqa Applied University at Jordan since 2011. Dr. Obaida received his PhD in Computer Science/ Cryptography from Malaysia in 2010. He co-authored 2 book chapters, and 27 research articles in leading ISI / International refereed journals. E-mail: dr_obaida@bau.edu.jo