3D Visualization for Concept Location in Source Code

Xinrong Xie
Department of Computer Science
Wayne State University
Detroit, MI, 48202
xxr@wayne.edu

Denys Poshyvanyk
Department of Computer Science
Wayne State University
Detroit, MI, 48202
denys@wayne.edu

Andrian Marcus*
Department of Computer Science
Wayne State University
Detroit, MI, 48202
amarcus@wayne.edu

ABSTRACT
The paper presents a set of tools that work in conjunction to support concept location in software. One of the tools, IRiSS (Information Retrieval based Software Search), is a search engine, designed and implemented to allow searching the source code of a software system. The other tool, sv3D (source viewer 3D), is a visualization front end, designed to represent software data with 3D renderings.

The two tools are integrated with MS Visual Studio, with IRiSS providing the infrastructure for indexing the source code and querying, while sv3D helps the user in visually navigating the results of the queries and keeps track of the navigation path.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**]: User interfaces; D.2.6 [**Programming Environments**]: Graphical environments, Interactive environments.

General Terms
Design, Management

Keywords
Software Visualization, Concept Location, Information Retrieval

1. INTRODUCTION
The main goal of concept location is finding the parts of the source code that implement specific domain concepts. Concept location is a frequent software engineering activity that supports various software maintenance and evolution tasks such as incremental change and reverse engineering. The two major classes of concept location techniques are known as static [9] and dynamic [17]. While dynamic techniques require execution traces and test suites, static techniques are based solely on source code. The main instances of static concept location are regular expression matching, dependency search [2], and informational retrieval (IR) techniques [10].

One of the issues, common to all current implementations of the three techniques for static concept location, is the lack of an overview of the search results within the context of the whole software system and relationships between the source code elements present in the results. A natural solution to this problem is the visualization of the parts of the source code that are related to the concept. However, there is little effort done in trying to support these techniques with visualization. Most of the existing approaches are tailored to the visualization of dependency graphs. One of the problems in visualizing the results of a string matching search, for example, is the variable granularity and the unanticipated size of the results of a given search.

The presented tools offer solutions on how to use visualization during the process of concept location. Visualization can enrich the representation of the relevant search results showing their locations in the source code, as well as their density distribution within the context of the complete software system. Visualization can also be used to capture and show the user's browsing history for multiple queries, to represent a cross-query overview of the results.

The first tool, IRiSS [11], implements an information retrieval (IR) based concept location method as an add-on for MS Visual Studio .NET. The other tool, sv3D – source viewer 3D [8], is a visualization front end used to represent software data with 3D graphical metaphors.

2. RELATED WORK
A number of visualization tools were developed to support program comprehension, especially concept or feature location. TraceGraph [7] supports concept location tasks where particular features of a program are implemented in order to fix a bug or introduce an enhancement, particularly for large, long-running or interactive software. The tool provides a simple visual display of the program traces which allows changes in execution to be easily distinguished. Relo [15] builds and automatically manages a visualization mirror of the developer's mental model, allowing them to group viewed artifacts or use the viewed items to ask the system for further exploration suggestions. FEAT [13] allows the programmer to create views of structurally related elements by adding them to concern graphs, which can be associated with the source code through a query mechanism.

Some of the existing visualization tools support browsing and searching in the source code. SHriMP [16] and Creole [6] allow users browsing the source code and documentation via

ICSE'06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

*Andrian Marcus is currently visiting in the Department of Mathematics and Computer Science at the "Babeş-Bolyai" University, Cluj-Napoca, Romania

embedded hypertext links through combining code and structural browsing. They also support three searching strategies: general search, artifact search and relation search. Aspect Browser [4] uses regular expressions to locate specific artifacts and visualize their distribution. CodeSurfer [1] provides a wide range of program understanding capabilities by exposing the results of a static-semantic analysis to the user. It allows searches for all the references given variables and performs sophisticated impact analysis. NavTracks [14] supports browsing software by keeping track of the navigation history of software developers, forming associations between related files. These associations are used to recommend potentially related files.

None of the existing software visualization tools support IR style queries with multiple terms, as used in most current search engines. This provides the main motivation behind combining sv3D and IRiSS.

3. SOURCE VIEWER 3D

Source Viewer 3D (sv3D) is a software visualization framework that uses a 3D visual metaphor, which extends the SeeSoft [3] file and pixel maps. In particular, sv3D creates 3D renderings of the software data based on user defined mappings. It uses poly cylinders to represent software entities (e.g., characters, lines of code, methods, functions, classes, etc.) and it uses containers to show aggregates of these elements (see Figure 1 and Figure 2). The height and color of the poly cylinders is used to represent various software metrics or other data relevant to the software system.

sv3D was designed to interact easily with different tools that can perform various analysis on the source code. It supports advanced user interaction techniques and uses the 3D space for visualization. For example, the overview feature is able to show efficiently large amounts of source code in one view. sv3D also allows zooming and panning at variable speeds; moreover, two types of 3D manipulators (i.e., track ball and handle box) are available to the user to interact with the visualization, at object level. In addition, a number of filtering methods are defined (e.g., transparency), which allow the user to view only information relevant to their needs. The tool also allows the user to take snapshots of the current view.

Overall, sv3D can be used to support tasks such as source code searching and browsing, impact analysis, evolution, slicing, fault localization, visualization of execution traces, etc.

4. CONCEPT LOCATION WITH IRISS

Concept (or feature) location is one of the most frequent activities that software developers perform when there is a need to change something in the existing software system. It is routinely done during incremental change of the software [12]. An overview of different methods is available in [9, 17]. Here, we concentrate on the IR-based [10] concept location method, implemented in the IRiSS tool [11], which is built as an add-on for the Microsoft Visual Studio .NET development environment.

In a nutshell, the concept location process using IRiSS has the following steps: (1) preprocessing the source code and documentation; (2) building a vector space representation of a software system using Latent Semantic Indexing (LSI) [5]; (3) executing queries formulated by the user; and (4) retrieving and analyzing the results, which are returned as a ranked list of source code elements depending on the chosen level of granularity (e.g., method or class level).

Steps 1 and 2 are usually done once automatically, while steps 3 and 4 are performed iteratively by the user during concept location. IRiSS uses the MS Visual Studio interface to represent search results in text format (e.g., method prototypes, location, and similarity measures to the user query).

5. MOTIVATION FOR MERGING SV3D AND IRISS

There are several reasons why the combination of sv3D and IRiSS is beneficial for different program comprehension activities. First, IRiSS' interface, as in most searching tools, is text-based. One problem with such an interface is that it does not provide the user with an overview of the results for searches and the relationships among identified source code elements. Visualization can solve this problem by providing a visual map of the source code elements based on the similarity measure to a query as well as relationships among the results. The distribution of the results may be important for various comprehension tasks and it is usually lost if simple textual representation is used. Moreover, filtering of the results, represented in a visual form, may be more natural for users.

In addition, during concept location or a simple set of searches, the user needs to keep in mind more than just the results of the latest query. It is important, for example, to see if a certain method or class was returned as a result for one or more previous queries, since the process of formulating a good query may take several iterative steps. Textual representations may present difficulty in showing this type of information, but visualization would help to preserve this information represented using several visual attributes of the used metaphors.

As methods and classes in the software system are investigated, it is often more useful to have additional information about these classes (e.g. metrics) than just the source code. sv3D can display this kind of information which can be mapped to color or height of the ploy cylinders, together with the results of a search.

Finally, integrating sv3D with one or more development environments is one of our original goals in its development. Since IRiSS is integrated into the Visual Studio .NET, it provides a good communication mechanism between the IDE and sv3D.

6. SEARCHING AND BROWSING WITH IRISS AND SV3D

In the typical usage scenario envisioned by us, after IRiSS generates results for a user query, they are transferred to sv3D for visualization based on the user preferred mappings. Then, the user can explore the results either in the IDE or in sv3D and navigate between the two. While exploring in sv3D, the user is able to double-click on a poly cylinder, which represents a method, and navigate directly to the corresponding line of source code in the text editor.

6.1 Using Colors to Express Similarities

IRiSS is meant to complement or replace the existing find feature in MS Visual Studio. The granularity of the search results in IRiSS may be at method or class level (i.e., IRiSS returns methods or classes as results to a user query, rather than lines of text). The search results are displayed in the standard output window in Visual Studio sorted in decreasing order based on similarity values between the query keywords and the respective methods. The format of the results includes method name, path, line of code where implementation for this method starts, and the similarity with a query [11].

Although it is obvious how to explore ranked results displayed as a table, in this case the users may be unaware of the other important attributes of the method, the class, the package or even the whole software system. Thus, besides the text view of the results in the IDE, the user may wish to have access to other views in sv3D (i.e., the display of various analysis data).

In our approach we decided to use colors for visualizing the similarities, based on a pre-defined color scheme. The similarity measure is defined as the cosine between source and target vectors in the LSI space [10] and it is in the range of [-1, 1]. The similarity measure is mapped to the color of the poly cylinders, which represent methods in each class in the order as they appear in the source code.

Since different users may have different preferences for colors, they are able to define their own color scheme using sv3D properties. By using colors to represent relevance of the query results it is easy to identify, for example, the most relevant methods in the class as well as the overall number of relevant methods in that particular class (see Figure 1). In addition, the user may get more information about the method represented in sv3D by clicking on the respective poly cylinder. When it is hard to differentiate between two poly cylinders of the same color, the user may simply click on the poly cylinder to see the similarity value in the control panel or double click on the poly cylinder in sv3D and return to IDE to the exact location of the corresponding method.

Figure 1. Using colors to express similarities to the query.

6.2 Using Height to Display Browsing History

Maintenance of large software projects always requires programmers to explore the source code frequently. The good support for these types of activities is necessary. One of the key issues with many existing tools for browsing and exploration is the lack of support for history of the browsing steps for backtracking during the concept location process. One of the solutions to this problem is to keep track of the browsing history so the programmer has additional information about the browsing process including visited methods, number of times each method has been explored, and the exploration sequence of the methods.

Using the third dimension in sv3D allows the visualization of the browsing history through the height of the poly cylinders. Every time the programmer explores a method, the height of the poly cylinder will increase by one unit, and the exploration sequence will be displayed on the poly cylinder (see Figure 2).

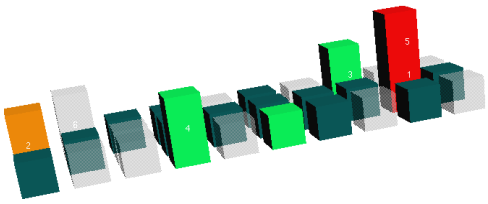

Figure 2. Using height to visualize browsing history.

6.3 Combining IRiSS and sv3D

One of the reasons that software visualization tools are not widely used in software development is that most tools are independent of the development environment and can not be easily used during the development and maintenance process. Although a tool may be helpful in program comprehension, there may be quite a bit of overhead to run the visualization tool and use it separately from the development process. In other words, successful tools should be developed as plug-ins to IDE or tightly coupled with IDE.

Thus, we decided to integrate IRiSS and sv3D by implementing message passing between the tools. Whenever the user runs the first query in IRiSS, sv3D in turn will get the notifying message and visualize the results of the query. In turn, whenever the user explores a method in sv3D, he or she may double-click on a poly cylinder and return to IDE. IRiSS in this case will receive the message from sv3D to show that particular method in the output window and display the implementation of that method in the editor. The process of using IRiSS together with sv3D is outlined in Figure 3.

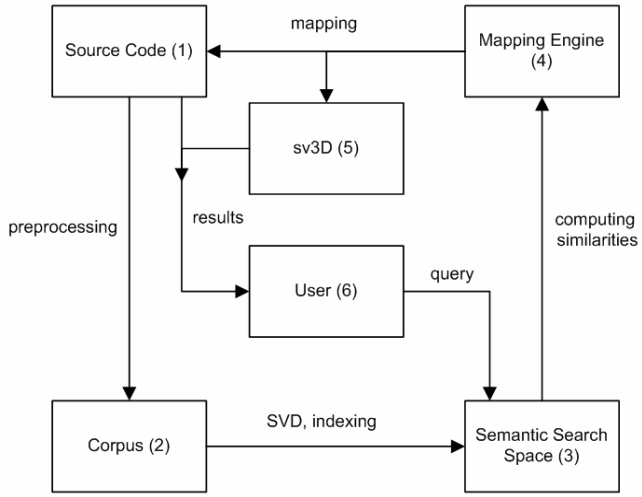

Figure 3. Workflow for concept location with IRiSS & sv3D.

Overall, the bi-directional interaction between IRiSS and sv3D makes it possible for the programmer to easily switch between IDE and the visualization window. From a hardware point of view, the tools are optimized to work with dual monitor displays, one showing the IDE and the other the visualization.

7. CONCLUSION AND FUTURE WORK

We believe that enriching source code searching and browsing with visualization support will improve concept location and other similar activities. The experience of merging two technologies and tools, IRiSS and sv3D, has promising results.

Recording and visualizing historical data about searching and browsing activities can be useful for software engineers. As future work, we plan to experiment with different dynamic mappings as well as to include static information on dependencies and other information to improve the location process. User evaluations are needed and planned as well.

Overall, IRiSS and sv3D should be more tightly coupled; for example, one could use sv3D to select a method in the system and use it as a query in IRiSS.

8. AVAILABILITY

IRiSS and sv3D are free for academic use. Please contact the authors for the most recent version of the tools.

9. ACKNOWLEDGMENTS

We are grateful to Yubo Dong, Andrey Sergeyev, Louis Feng and Jonathan Maletic, who contributed to the development of the previous versions of IRiSS and sv3D. This research was supported in part by grants from the National Science Foundation (CCF-0438970) and the National Institute for Health (NHGRI 1R01HG003491).

10. REFERENCES

[1] Anderson, P. and Zarins, M. The CodeSurfer Software Understanding Platform in Proceedings of 13th International Workshop on Program Comprehension (IWPC'05) (St. Louis, Missouri, USA, 2005), 147-148.

[2] Chen, K. and Rajlich, V. Case Study of Feature Location Using Dependence Graph in Proceedings of 8th IEEE International Workshop on Program Comprehension (IWPC'00) (Limerick, Ireland, June 2000), 241-249.

[3] Eick, S., Steffen, J. L., and Summer, E. E. Seesoft - A Tool For Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering*, 18, 11 (November 1992), 957-968.

[4] Griswold, W. G., Yuan, J. J., and Kato, Y. Exploiting the Map Metaphor in a Tool for Software Evolution in Proceedings of 23rd IEEE International Conference on Software Engineering (ICSE'01) (Toronto, Ontario, May 12-19, 2001), 265-274.

[5] Landauer, T. K. and Dumais, S. T. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104, 2 (1997), 211-240.

[6] Lintern, R., Michaud, J., Storey, M.A., Wu, X. Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse in Proceedings of Proceedings of the 2003 ACM Symposium on Software Visualization (SoftViz'03) (2003), 47 - 57.

[7] Lukoit, K., Wilde, N., Stowell, S., and Hennessey, T. TraceGraph: Immediate Visual Location of Software Features in Proceedings of 16th IEEE International Conference on Software Maintenance (ICSM'00) (Washington DC, USA, 2000), 33-39.

[8] Marcus, A., Feng, L., and Maletic, J. I. 3D Representations for Software Visualization in Proceedings of 1st ACM Symposium on Software Visualization (SoftVis'03) (San Diego, CA, June 11-13, 2003), 27-36.

[9] Marcus, A., Rajlich, V., Buchta, J., Petrenko, M., and Sergeyev, A. Static Techniques for Concept Location in Object-Oriented Code in Proceedings of 13th IEEE International Workshop on Program Comprehension (IWPC'05) (St. Louis, Missouri, USA, 2005), 33-42.

[10] Marcus, A., Sergeyev, A., Rajlich, V., and Maletic, J. An Information Retrieval Approach to Concept Location in Source Code in Proceedings of 11th IEEE Working Conference on Reverse Engineering (WCRE'04) (Delft, The Netherlands, November 9-12, 2004), 214-223.

[11] Poshyvanyk, D., Marcus, A., Dong, Y., and Sergeyev, A. IRiSS - A Source Code Exploration Tool in the Industrial and Tool Proceedings of 21st IEEE International Conference on Software Maintenance (ICSM'05) (Budapest, Hungary, September 25-30, 2005), 69-72.

[12] Rajlich, V. and Wilde, N. The Role of Concepts in Program Comprehension in Proceedings of IEEE International Workshop on Program Comprehension (IWPC'02) (2002), 271-278.

[13] Robillard, M. P. and Murphy, G. C. FEAT a tool for locating, describing, and analyzing concerns in source code in Proceedings of 25th International Conference on Software Engineering (ICSE'03) (Portland, OR, May 3-10, 2003), 822-823.

[14] Singer, J., Elves, R., and Storey, M.-A. D. NavTracks: Supporting Navigation in Software Maintenance in Proceedings of 21st IEEE International Conference on Software Maintenance (ICSM'05) (Budapest, Hungary, Sept 25-30, 2005), 325-334.

[15] Sinha, V., Miller, R., and Karger, D. Incremental Exploratory Visualization of Relationships in Large Codebases for Program Comprehension in Proceedings of 20th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'05) (San Diego, CA, 2005), p.198-199.

[16] Storey, M.-A. D., Best, C., and Michaud, J. SHriMP Views: An Interactive Environment for Exploring Java Programs in Proceedings of Ninth International Workshop on Program Comprehension (IWPC'01) (Toronto, Ontario, Canada, May 12-13, 2001), 111-112.

[17] Wilde, N., Buckellew, M., Page, H., Rajlich, V., and Pounds, L. A Comparison of Methods for Locating Features in Legacy Software. *Journal of Systems and Software*, 65, 2 (February 15 2003), 105-114.