Visual Information Discovery in Software Systems

Peter Kapec, Martin Šperka

Faculty of Informatics and Information Technologies,
Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava
{kapec, sperka}@fiit.stuba.sk

Abstract. Visual and interactive immersion into multidimensional and dynamic data helps to discover hidden structures and relations. In this paper we concentrate on visual information discovery in large structures and propose methods for creating and rendering hyper-graph representations of software artifacts.

Keywords: information visualization, software mining, software artifacts

1 Introduction

Knowledge discovery is an interactive and iterative process. It is important to involve the user into this process. Visual data mining aims to help the user with the data exploration by providing visual insight into data. Visual representations allow interactive and intuitive examination of inhomogeneous data and do not require that the user understands difficult principles of automatic data mining techniques. Of course automatic data exploration allows dealing with huge data sets due to computer's computational power, but visual exploration adds user's knowledge and creativity and exploits large bandwidth of human visual channel.

In last time multidimensional, dynamic and abstract data like source code, data structures, databases, algorithms etc. are in the focus of interest. Studies done by Petre et al. show that graphical representations may offer following advantages [7]:

- Better information content and information density
- Higher level of abstraction
- Providing overview (structures may become easier recognizable)
- Images are good communication instruments, they may over-bridge language barriers
- Images are handily graspable and easily memorable

Visual exploration techniques can be classified according to three orthogonal criteria [3]: data to be visualized, *visualization techniques* and *interaction techniques*. *Data type* may be n-dimensional data, text and hypertext, hierarchies and graphs, temporal data, algorithms and software, which become very important today. Many visualization techniques were developed to deal with software and a large part of them are based on graph visualizations.

2 Software mining and visualization

Under knowledge we often understand information placed into a meaningful context, combined with experience, interpreted etc. Ontologies allow representing and storing knowledge about a certain domain in a declarative way and can be interpreted by humans and also computers. Of course, presenting ontologies in a navigable form is important and often visualizations based on graphs are used. Graphs, or even hypergraphs, can be used to formally define two popular standards for ontology representation, Topic Maps and RDF/OWL.

Knowledge discovery has many application areas. One of them, software mining, deals with understanding software artifacts. The term software covers many different things ranging from source code, documentation, source code revisions, diagrams, graphical user interfaces, data and algorithms and many others. It is difficult to deal with such different software artifacts, especially in large projects. Software mining and software visualization aim to simplify extraction of previously unknown and potentially useful information from software. An interesting idea is to store knowledge about different kinds of software artifact using one representation [8]. Topic maps, a standard for ontology representation, allow representing knowledge about software artifacts. Topics can be used to represent information about all kinds of software artifacts and associations can represent different relations between them.

It seems that software comprehension is easier when the software is graphically represented. In the past two decades many software visualization prototypes and visual programming environments were developed. However only few managed to move from research projects to systems usable in praxis. Results of Koschke's study in the software visualization field lead to following challenges [4]:

- visualization of large datasets and dynamic aspects
- navigation between multiple views
- automatic selection of visualization techniques
- semantic visualization
- integration with other tools

A possible research direction in the field of software visualization is to utilize standards for ontology representation, e.g. topic maps, which can be formally described through hypergraphs. A formal definition of a hypergraph follows.

Definition 1: *Let $V = \{v_1,...,v_n\}$ be a finite set, whose members are called nodes. A hypergraph on V is a pair $H = (V,\varepsilon)$, where ε is a family $(E_i)_{i\in I}$ of subsets of V. The members of ε are called edges.*

Hypergraph nodes can represent topics and hypergraph edges can represent associations of a topic map. Other topic map features can be modeled with jumps between semantic layers represented through connected components of a hypergraph [1]. Hypergraphs offer better expressiveness than graphs, because they allow defining a relation between multiple objects.

3 Our research

Our intention is to introduce the ideas of uniform hypergraph representation of software artifacts to the software visualization field. Combining hypergraph representations and powerful querying techniques, which allow SQL-like database queries upon hypergraphs, with already existing information visualization, navigation and mining techniques, we could create an interesting development environment. A hypergraph representation can cover software artifacts from different aspects of development – we can mention source code artifacts (classes, objects, functions etc.) and source code relations (class-objects relations, class inheritance, call graphs, method usage etc.), documentations and revisions and even other development aspects like project management information, logs, debugging information, program runtime visualization, program profiling etc. All these features of a software project could be described using one structure, a hypergraph, allowing to explore these software artifacts using relations stored in the hypergraph.

We are working on a visual layer on top of a scripting language Lua [2]. Software artifacts we plan to represent through hypergraphs are source code, its documentation and revisions. The Figure 1 shows possible visualizations of a source code and documentation fragment. The left part displays a visualization of different semantic layers of a hypergraph representation [6]. Each layer can be dedicated to different software artifacts, thus visually grouping them. The right part shows a hypergraph part in detail.

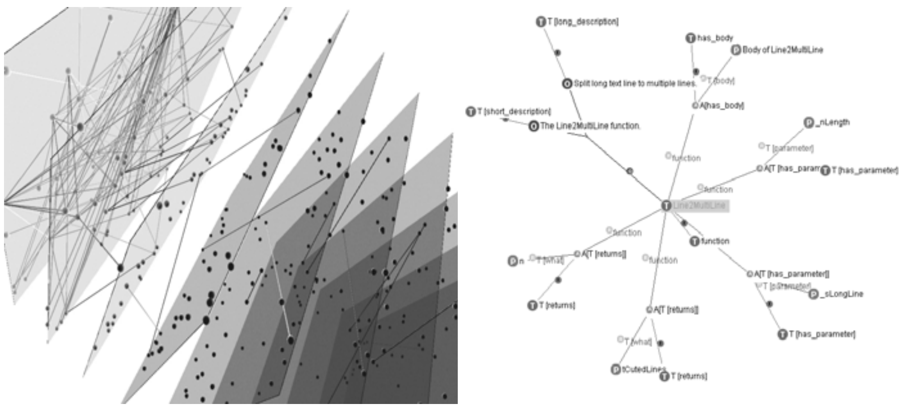

Fig. 1. Visualization of different kinds of software artifacts: using layers (left) and detail on hypergraph part (right).

Common graph visualization techniques are well suitable for source code comprehension as case studies done with CrocoCosmos [5] visualization system show. However, for programming tasks like modifying or writing new source code parts, we need access to concrete language constructs. Known zoom-interface techniques can be used.

4 Conclusions

Our goal is to develop a system, which will help programmers to understand and modify unknown source code and related software engineering task using intuitive and interactive graphical methods, which concentrates on semantic relations between various aspects of the project. In this paper we presented a possible approach in using topic maps to represent hypergraph representations of software artifacts.

Acknowledgement: This work was partially supported by the VEGA grant VG 1/3103/06: Information infrastructure for embedded knowledge processing in distributed environment.

References

1. Auillans, et al. A formal model for topic maps. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, Springer-Verlag, pp. 69-83, 2002
2. Celes, W., de Figueiredo, L. H., Ierusalimschy, R. Lua-an extensible extension language, In *Software: Practice & Experience,* Vol. 26, Num. 6, pp. 635-652, 1996
3. Keim D. Visual exploration of large databases, *Communications of the ACM*, Vol.44, Num.8, pp. 38-44, 2001
4. Koschke, R. Software Visualization in Software Maintenance, Reverse Engineering, and Reengineering: A Research Survey. In *Journal on Software Maintenance and Evolution*, John Wiley & Sons, Ltd., Vol. 15, No. 2, pp. 87-109, 2003.
5. Lewerentz, C., Simon F. Metrics-based 3D Visualization of Large Object-Oriented Programs. In *Proceedings of the First International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT02)*, 2002.
6. Manduch, J. Applications of Virtual Reality in Visual Data Mining. Master thesis. Faculty of Informatics and Information Technologies, STU Bratislava, 2006
7. Petre, M., Blackwell, A., Green, T. Cognitive Questions in Software Visualization. In: J. Stasko, J. Domingue, M. Brown, B. Price (Ed.): Software Visualization – Programming as a Multimedia Experience, pp. 453-480, MIT Press, Cambridge, MA, 1998.
8. Rauschmayer, A., Renner, P. Knowledge-Representation-Based Software Engineering. Technical report TR-0407, Ludwig Maximilian University, München, 2004