Towards End-User Web Software Visualization

Craig Anslow, James Noble, and Stuart Marshall
School of Mathematics, Statistics,
and Computer Science
Victoria University of Wellington, New Zealand
{craig, kjx, stuart}@mcs.vuw.ac.nz

Ewan Tempero
Department of Computer Science
University of Auckland, New Zealand
ewan@cs.auckland.ac.nz

Abstract

Software visualization has always been expensive, special purpose, and hard to program. Most of the existing software visualization tools require too much time for end-user developers to learn and make effective use of. We are currently building a web software visualization application that allows end-user to create, view, save, and share visualizations. In this abstract we introduce our software corpus visualization project and summarize our results thus far.

1. Introduction

We are interested in understanding what software looks like to help with software reuse, software maintenance, and software re-engineering. We believe creating software visualizations will help to assist end-user developers to understand the structure and behaviour of software. We want to create visualizations of object-oriented programs over the web cheap, portable, easy, and usable for end-user software developers. We have a corpus of Java software[1] that contains 94 open-source Java applications, 22 applications with multiple versions, with 233 versions total. The corpus is used for conducting empirical studies to help understand how software engineers create code and the relationship between the code structure and quality attributes such as modifiability, reusability, maintainability, and testability. Our project requires tools for visualizing the software.

Software visualization has always been expensive, special purpose, and hard to program. Our previous work in developing a web-based software visualization architecture [5] has explored creating visualizations with Scalable Vector Graphics (SVG) [3] and Extensible 3D Graphics (X3D) [1]. We are building a web software visualization application that makes uses of these technologies, and allows end-users to upload Java files and then create different visualizations. The rest of this abstract looks at existing software visualization tools, characterizes web information visualization tools, and discusses implications for our research.

2. Web Software Visualization

In a recent survey [4] based on questionnaires completed by 111 researchers from software maintenance, reengineering and reverse engineering, 40% found software visualisation absolutely necessary for their work and another 42% found it important but not critical. The majority of the researchers are primarily using or integrating existing software visualizations tools developed by others (33%). The survey did not ask what kind of software visualization tools were used.

We want software visualization to be an easy task for end-users without the need for downloading and installing separate applications. However, it is not clear what a good software visualization system looks like. We believe the web is an excellent platform for creating a software visualization application. Web based software visualization allows end-users to independently create, view, save, and share visualizations with others.

We have explored the 43 software visualization tools listed by Diehl [2] and found that only one of the tools was web-based, the SHriMP[2] application for visualizing dependencies in hierarchically structured data as nested graphs. The rest of the applications require a separate download, plug-in to an IDE such as Eclipse, or are proprietary software. Since there is a lack of freely available web software visualization tools we have decided to explore existing information visualization web tools. We want to see if any of these tools have useful features that we could incorporate into our software corpus visualization project.

Many Eyes[3] is a web site that provides collaborative vi-

[1] http://www.cs.auckland.ac.nz/~ewan/corpus/

[2] http://www.thechiselgroup.org/shrimp

[3] http://www.many-eyes.com

sualization services. Figure 1 shows two visualizations we created. The visualizations show the words used in the classnames from the Java Standard API specification version 6. For example AbstractColorChooserPanel becomes Abstract (position one), Color (two), Chooser (three), and Panel (four). The tag cloud shows the most common words used in Java classnames are Basic (75 occurrences), Metal (54), Default (51), Invalid (50). There are 1217 unique words in position one, 761 in position two, 409 in three, 186 in four, and 70 in five. The treemap shows the order of the words in the Java classnames and the most prominent word in position one is Metal followed by Basic, Default, Order, and Key.

Other information visualization web tools include Swivel[4], Data360[5], and DataPlace[6]. These other tools operate similar to Many Eyes but provide less sophisticated visualizations. There are some key characteristics of all of these web visualization tools. First, they require end-users to register with the web site. Second, an end-user can upload data in ASCII or spreadsheet format. Third, end-users can modify their data online. Fourth, multiple visualizations can be created from the data and at any time. Finally, end-users can comment on the visualizations.

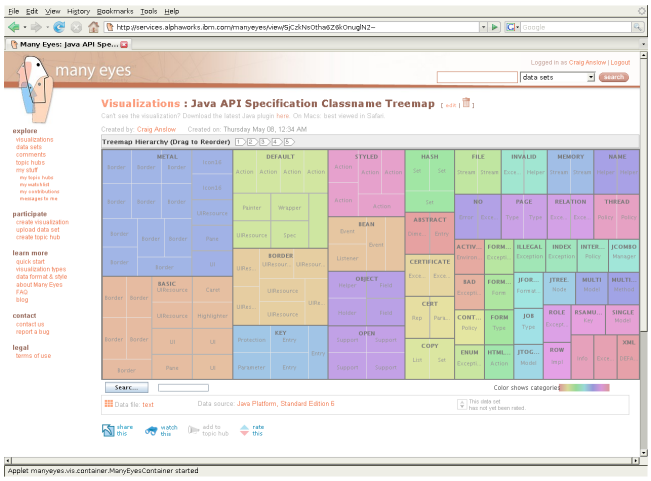
Instead of uploading ASCII text we want to be able to upload Java source code and .jar files from our corpus. We would like to incorporate some of the information visualization techniques used in web visualization tools (e.g. tag cloud and treemaps) and specific software visualization techniques including: UML diagrams, algorithm animations, metric visualizations (e.g. class blueprints, polymetric views), and software evolution (e.g. Seesoft-like, software archives). Including social features such as end-user registration and commenting would be useful assets of the application for supporting open-source development.

There are some implications for our research. We need to determine what visualization types to implement. Depending on the visualization type different methods will be required to parse the source code or reverse engineer the .jar files. We intend to make our software visualization system public facing so we will need to consider how we handle proprietary software and how the system scales once there are lots of software uploaded and many visualizations created. Once our application is in production we intend to conduct user evaluations (user testing and interviews) to see how effective the system is.

In summary, there is a lack of easy to use web software visualization systems. We are working towards a web based application that will help end-users to upload their Java software applications, create visualizations, and share their visualizations with other users.

[4]http://www.swivel.com
[5]http://www.data360.org
[6]http://www.dataplace.org

(a) Tag Cloud



(b) Treemap

Figure 1. Many Eyes - visualizations of the words used in the classnames from the Java API Specification.

References

[1] C. Anslow. Evaluating X3D for use in software visualisation. Master's thesis, VUW, 2007.

[2] S. Diehl. *Software Visualization*. Springer Verlag, 2007.

[3] M. Duignan, R. Biddle, and E. Tempero. Evaluating scalable vector graphics for use in software visualisation. In *Proc. of INVIS*, 2003.

[4] R. Koschke. Software visualization for reverse engineering. In *Revised Lectures on Software Visualization*, pages 138–150. Springer Verlag, 2002.

[5] S. Marshall, K. Jackson, R. Biddle, M. McGavin, E. Tempero, and M. Duignan. Visualising reusable software over the web. In *Proc. of INVIS*, 2001.