

An Interactive Visualization Tool for Understanding Complex Programs^{*}

J. CHERRY, M. ARRIETA, E. BROWN AND S. RAMASWAMY

*Software Automation and Intelligence Laboratory, Department of Computer Science and Center for Manufacturing Research
Tennessee Technological University, Cookeville TN 38505. Phone: (931)-372-3691 Email: srini@acm.org / srini@ieee.org*

Abstract: In this paper, we present the development of a program parser coupled with an off-the-shelf interactive program visualization tool that assists in the understanding of complex computer programs. The major objective of this work is to quickly help newcomers become familiar with existing software code and become productive members of the software development team. These software systems that often incorporate several interacting pieces will be produced and maintained by several individuals and/or vendors. Hence, it will be difficult for one individual or group to be fully cognizant of the intricacies of the overall software system. In work-related training, interactive visual systems will be required to provide support for teaching organizations' newer workforce to quickly learn and effectively contribute to the development and maintenance of these complex and critical software systems. An environment that provides an in-depth view of the software system will be necessary to fulfill this requirement. Additionally, an interactive visual system may provide a means to dynamically access and analyze the skills and capabilities of trainees for skill-based job placement.

I. Introduction

Becoming familiar with a complex software program is an intricate and time-consuming task for individuals and group members new to a particular project. The task can be somewhat simplified with the use of a graphical visualization tool that displays a structural view of the software system. The hierarchical structure of programming code and its components can be represented in numerous ways through the use of various visualization environments.

This paper is organized as follows: Section II presents an overview of the various visualization tools commonly found in the literature. Section 0 presents our general framework for program visualization tools. Section IV presents an example visualization parser built for C++ programs. Section V concludes the paper.

II. Background

In the process of development of a tool for visual presentation of software programs, it is necessary to select a visualization tool that is capable of presenting large programs in a form that is easily visualized and traversed. In this section, we will first present the various visualization tools found in the literature and the major drawback of these tools, and then present several commercial hyperbolic tree based visualization tools. In our approach we chose the hyperbolic visualization mechanism because of its widespread commercial adoption.

II. A. Overview of Visualization Tools

Several researchers have worked on visualization environments for software systems. In [1-2], the authors provide a Java based visualization tool with a zoom-able interface. This tool provides context-based visualization, which is a semantic presentation that displays a particular view of a node based on a specific task. It presents a very flexible visualizer that is capable of three different types of zooming – geometric, semantic and fisheye. In geometric zooming the user is able to scale a nested view around a central point. In semantic viewing the user is able to display specific semantics about the selected node in the visualization – for example, visualizing children nodes, documentation, etc. However, it is developed for visualizing Java programs. In [3] the authors present Tree maps, a tool for quick overview of hierarchically structured data. While Tree maps effectively utilize limited space, they offer no navigational capabilities, which are essential for a tool whose purpose is to provide visualization of software code. Other tree-based visualization tools include the Spence layout [6], fractal approaches [7], cone trees [8], and multi trees [9]. Issues addressed in these various mechanisms include: efficient usage of screen space, presentation elegance, and clarity to the user.

^{*} This work was carried out at the Software Automation and Intelligence Laboratory in the Department of Computer Science at Tennessee Technological University as part of our research on intelligent coordinating entities, or ICE. Dr. S. Ramaswamy is an Associate Professor and Chair of the Computer Science Department at Tennessee Technological University. Phone: (931)-372-3691. Email: srini@acm.org / srini@ieee.org.

representation, and the hyperbolic visualization systems appear to be ideal for program understanding; wherein program dependency structures are often unbalanced. It has been noted that by trying to provide a rotating structure such as a hyperbolic tree, users have been able to gain useful insights into the information being visualized [8]. This is due to the way in which programmers perceive the structure and interconnections of the components of a modular software system. Often the users, usually new comers to a

programming team, are people who already have a good sense of the programming language, which helps them in understanding the context of the information being presented.

III. General Framework

One of the main objectives of our work is the development of a full-fledged visualizing environment that is not specific to a particular language or environment. Thus we started off

```
class C_no : public B_no {
public:
    int i;
};
class B_no {
private:
    int j;
};

class Point {
private:
    Point() {}
};

class Circle:Point{
public:
    int i;
};

class C_yes : public B_no {
public:
    int value;
};
class b_no {
public:
    int stuff;
};
```

Figure 3. Sample Program Fragment

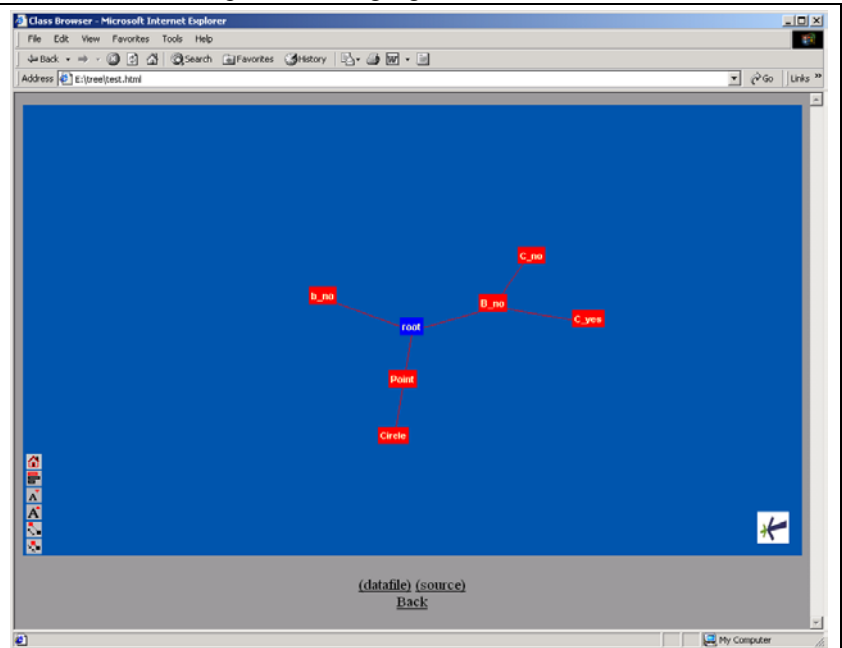


Figure 4. Basic View

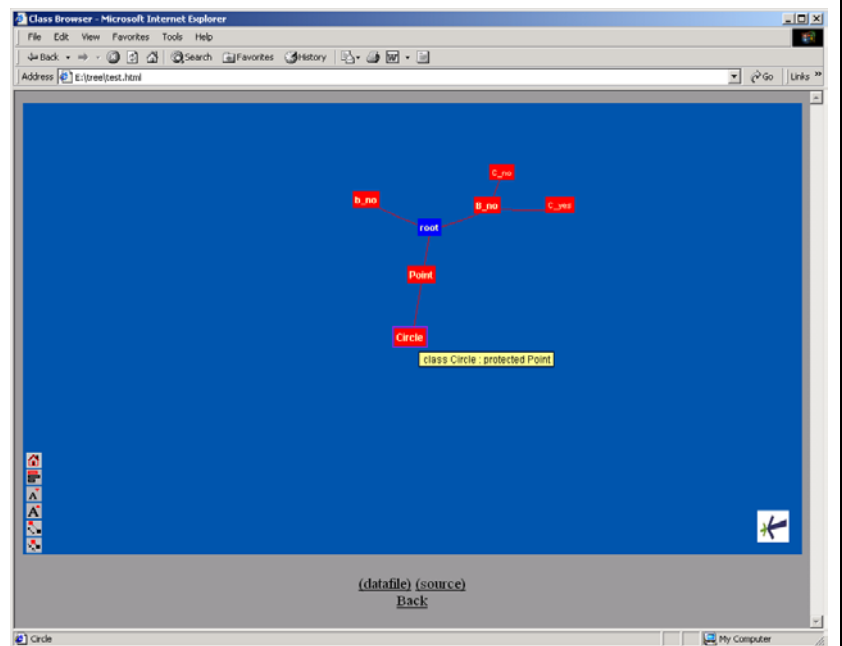


Figure 5. Visualizing Class Derivation Information

