Chapter1 Introduction

In this thesis, we develop one tool, YaJDB, to provide debugging information for DIVINE and the necessary support for animation metaphor to makes it possible to animate general-purpose programs.

1.1 Overview

Debugging is the critical part of software development. One primitive way to debug a program is inserting print instructions into a program to print out variable values for detecting bugs. This way is helpful but requires a substantial amount of programmers' time and effort. Hence programmers need a tool to debug more effectively without spending much effort. This tool is called debugger.

Debuggers are software tools that help programmers to understand a program and find the cause of the defects more effectively. Formally, a debugger is a tool that controls the execution of application being debugged, so that programmers can inspect the program execution, stop the program, and probe the state of the program to verify its correctness.

Nowadays, source-level debuggers are a standard part of every programming environment. Most of them provide a rich suite of debugging facilities such as setting breakpoints, stepping line by line, watching variable values and so on. Many debuggers provide graphical user interface (GUI) which allows users to use mouse

to invoke these facilities, such as setting a break point by a mouse click.

Even if debuggers are of great helpful for debugging, programmers still spend considerable time in debugging than coding. The difficulty of debugging depends on the complexity and size of a program. A small scale program is easier to understand than a large one and the impact of changes can more easily be traced. A large program is not easy to understand and trace even if it is documented well. Therefore, the key problem of debugging is how to understand a program, particularly if the program is not written by person who is responsible for debugging.

Traditionally, programmers understand a program by tracing the program with a debugger or by reading related documents. However, these methods are not very effective for a complex or large program. In essence, the traditional methods base on the textual presentation, but human beings receive information more easily though pictures. Hence, graphical presentation could be easier and better to understand behavior and structure of a program than textual presentation.

There are many software visualization researches nowadays. Software visualization is an aid for program understanding, debugging and teaching of algorithms. Program animation is a kind of the software visualization that is concerned with dynamic graphical displays of a program’s execution. But most of them were designed for specific algorithm or data structure. Only the few[14, 15, 4]

were designed for general purpose programs. However, like *Leonardo*[4], although it is one general-purpose program animation system, completing animation requires extra work from the users. A developer not only writes a program but is also responsible for designing how to animate a program under *Leonardo*. In our opinions, such tool does not have good usability.

In this thesis, we try to make it possible to animate general-purpose programs without extra efforts. Therefore, we develop a tool called YaJDB to generate the execution history stored in XML files. In the future, animation metaphor programmer will design animation metaphors to render different kinds of execution history based on our support. That is, we attempt to provide support independently from animation metaphors, which may render an execution history differently. Besides, YaJDB also provide essential support for DIVINE[5, 6, 16].

1.2 Organization

The remainder of the thesis is structured as follows: next chapter describes related background knowledge, including debugger, software visualization, XML and related technology. Chapter 3 introduces the YaJDB and how to support DIVINE. Chapter 4 describes how YaJDB support general-purpose program animation system. And we implement one experimental program animation tool to prove information YaJDB produces work in chapter 5. Finally, chapter 6

summarizes the conclusions and future works.