Program Execution and Visualization on the Web

Cristóbal Pareja-Flores
Universidad Complutense de Madrid, Spain

J. Ángel Velázquez-Iturbide
Universidad Rey Juan Carlos, Spain

INTRODUCTION

Programming is a central activity in the computing profession. It is facilitated by different tools (editors, compilers, debuggers, etc), which are often integrated into programming environments. Programming also plays a central role in computer science education. For this purpose, a number of complementary tools were developed during the last decade: algorithm animators, program visualizers, problem generators, assignment graders, and so forth.

After the Web explosion, teachers of programming rapidly turned their attention to the Web. Although the Web has speed and power limitations, it also has several advantages that make it invaluable for educational purposes. Mainly, it provides universal accessibility and platform independence, and solves the distribution problem by always making available the last version of any tool.

This article describes Web-based tools for program execution and visualization (Pareja-Flores & Velázquez-Iturbide, 2002) in two sections. Each section gives a brief overview of their evolution, describes educational uses, and, in the case of visualization, includes some definitions and reports of lessons learned. Finally, we outline our view of future trends in the use of the Web for programming education and our personal conclusions.

BACKGROUND

The simplest use of the Web for programming education is as a public repository of high quality problems. Many collections have no structure or, at best, are lineally or hierarchically structured. We note several initiatives hosted by the ACM: the *Lab Repository* (Knox, 2002), *Computer Science Education Links* (McCauley, 2001), and the *ACM International Collegiate Programming Contest* (Skiena & Revilla, 2003). Other useful resources are also collected on Web sites, such as slides and audio lectures (Skiena & Revilla, 2003), algorithm animations (Brummond, 2001; Crescenzi et al., 2003), or programming tools (English, 2001). More advanced repositories provide a management system that, using (semi)structured mark-up languages, allows retrieving, maintaining and publishing. Good representatives are eXercita (Gregorio-Rodríguez et al., 2001, 2002) and SAIL (Kovourov, Tamassia, Bridgeman & Goodrich, 2000).

A more complex initiative consists in porting programming tools to be executed on the Web. We focus on two kinds of tools. The first kind is aimed at supporting program execution. Programming is a task that involves several activities: editing, compiling and testing, at least. These activities require running a number of applications: an editor, a compiler and the program itself. In addition to their conventional use in standalone computers, they can be used on the Web in a number of ways, increasing flexibility from an educational point of view.

A second kind of tools supports software visualization. This field studies the visual representation of software entities (Stasko, Domingue, Brown, & Price, 1998). Visualization requires an effort to abstract the target entity to visualize and to make a good graphical design that may yield many different representations: text vs. graphics, level of abstraction, static vs. dynamic visualizations, one or multiple views, behavior vs. performance, errors, and so forth. Algorithm animation is a subfield of software visualization aimed at the dynamic visualization a piece of software illustrating the main ideas or steps (i.e., its algorithmic behavior), but without a close relationship to the source code. The graphical nature of software visualization in general and algorithm animation in particular makes them very conducive to the hypermedia features of the Web.

Web-Based Program Execution

The simplest use of the Web for programming execution is as a medium to submit programs. A more advanced capability is the support of Web-based program edition, compilation and execution. This can be implemented in different ways, depending on how the client/server load is balanced. For instance, the system by Elenbogen,

Maxim and McDonald (2000) includes a set of interactive Web exercises on C++ delivered by Java applets. A different option consists in using the Web as the medium to transfer programs and data to the server, which is then responsible for compiling and executing programs. In this approach, programs may be edited by the client and then remotely compiled and executed on a server (Hiltz & Kögeler, 1997). The server may also work as a black-box tester based on input-output pairs (Arnow & Barshay, 1999; Skiena & Revilla, 2003).

There are few systems with full programming capabilities on the server side, including debugging. Ibrahim (1994) developed a system that allowed the programmer to use the Web as a front-end to edit a program, send it to the Web server, and debug it by performing tracing actions on the program running at the server. Finally, other systems give support to the graphical visualization of execution (Domingue & Mulholland, 1997) or allow the user to experiment by controlling which parts must be executed and how (Berghammer & Milanese, 2001).

Algorithm Animation

Algorithm animation is a research field which is now 20 years old and still evolving. There is a consensus with respect to the videotape Sorting Out Sorting presented in 1981 by Baecker (1998) a landmark on animation, which included animations of nine sorting algorithms. Afterwards, some works established the main techniques for specifying and implementing algorithm animation: BALSA (Brown, 1988), Tango (Stasko, 1990), and Pavane (Roman, Cox, Wilcox & Plun, 1992). Systematic categorizations of software visualizations have been proposed since then (Price, Baecker & Small, 1998).

In the mid-nineties, many of the existing systems were ported to the Web, and many additional systems were specifically designed for the Web. A representative work from these years is that of Naps (1996), in which he carried out a study of the technical alternatives that could be used to make animations produced by previous systems available on the Web. Other systems are JCAT (Brown & Raisamo, 1997), Mocha (Baker, Cruz, Liotta & Tamassia, 1995), and JHAVÉ (Naps, Eagan & Norton, 2000). A second group from these years is formed by the systems that were not specifically designed for the Web, but based on stand-alone multimedia and hypermedia. Although they could not be used directly on the Web, they and Web-based systems had similar hypermedia features. A good example is the HalVis system (Hansen, Schrimpsher & Narayanan, 1999).

Another category of systems automatically produce program animations, enhanced with graphical representations. These can be considered as extensions of programming environments with advanced animation capabilities which are usable on the Web. Three examples are ISVL for Prolog (Domingue & Mulholland, 1997), KIEL for ML (Berghammer & Milanese, 2001), Jeliot for Java programming (Haajanen et al. 1997) and WinHIPE for functional programming (Naharro-Berrocal, Pareja-Flores, Velázquez-Iturbide & Martínez-Santamarta, 2001).

EDUCATIONAL USES OF WEB-BASED PROGRAM EXECUTION AND VISUALIZATION

Program Execution

Currently, the most common use of the Web for programming courses is as a communication medium, facilitating submission and administration of assignments and grades (Burd, 2000). As we have described previously, it may also be used as a medium to retrieve exercises and to send, run and even debug programs at the server.

Web-based program execution is also being used in more specific ways within courses. First, if the Web interface of the programming tool is carefully designed, only one aspect can be considered at a time (the condition of a conditional statement, the body of a loop, etc.), allowing novice students to concentrate on program fragments that illustrate certain syntactic or semantic elements. This feature is especially important at the beginning, when novices ignore program structure and other details. Second, program execution on the Web can be used as a testing tool. For instance, the teacher can use the system for student inquiry by proposing that the student predict the behavior expected for a given input, or vice versa, by guessing the input that yields a desired output (Elenbogen et al., 2000). Finally, by using the Web to drive visualizations and animations (as explained in the next section), information can be given graphically to the programmer about errors, about the progress of the computation, and so on.

Algorithm Animation

From the outset, the main use of algorithm animation was educational, rather that industrial (for instance, as a debugging tool). Algorithm animation systems have been used in several ways: as a complement to lectures on algorithms, for self-study, or within laboratories. A more demanding use of animation systems consists in requiring students to build their own animations.

The best documented experience ran for about 20 years at the Computer Science Department of Brown University (Bazik, Tamassia, Reiss & van Dam, 1998). All