What are Visual Programming, Programming by Example, and Program Visualization?

Brad A. Myers
Dynamic Graphics Project
Computer Systems Research Institute
University of Toronto
Toronto, Ontario, M5S 1A4
Canada

ABSTRACT

There has been a great interest recently in systems that use graphics to aid in the programming, debugging, and understanding of computer programs. The terms "Visual Programming" and "Program Visualization" have been applied to these systems. Also, there has been a renewed interest in using examples to help alleviate the complexity of programming. This technique is called "Programming by Example." This paper attempts to provide more meaning to these terms by giving precise definitions, and then uses these definitions to classify existing systems into a taxonomy.

RESUME

Les systèmes qui utilisent l'infographie pour aider à la programmation, à la mise-au-point et à la comprehension de logiciels ont récemment suscité beaucoup d'intérêt. Les termes "programmation visuelle" et "visualisation de programmes" ont été associés à ces systèmes. Il y a aussi eu un renouveau d'intérêt pour l'utilisation d'exemples pour aider à simplifier la programmation. On parle alors de "programmation par exemples". Nous essaierons de définir ces termes avec plus de précision et utiliserons ces définitions comme base pour établir une taxonomie des systèmes disponibles actuellement.

Key Words and Phrases: Visual Programming, Program Visualization, Programming by Example, Inferencing, Automatic Programming, Flowcharts, Debugging Aids, Program Synthesis, Documentation, Computer Languages.

Extended Summary.

NOTE: This paper is a summary of [Myers 86]. The reader should refer to that paper for full information.

As the distribution of personal computers and the more powerful personal workstations grows, the majority of computer users now do not know how to program. They buy computers with packaged software and are not able to modify the software even to make small changes. In order to allow the end user to reconfigure and modify the system, the software may provide various options, but these often make the system more complex and still may not address the users' problems. "Easy-to-use" software, such as the "Direct Manipulation" systems [Shneiderman 83] actually make the user-programmer gap *worse* since more people will be able to use the software (since it is easy to use), but the internal program code is now much more complicated (due to the extra code to handle the user interface). Therefore, systems are moving in the direction of providing end user programming. It is well-known that conventional programming languages are difficult to learn and use [Gould 84], requiring skills that many people do not have. In an attempt to make the programming task easier, recent research has been directed towards using graphics. This has been called "Visual Programming" or "Graphical Programming". Some Visual Programming systems have successfully demonstrated that non-programmers can create fairly complex programs with little training [Halbert 84].

Another motivation for using graphics is that it tends to be a higher-level description of the desired actions (often de-emphasizing issues of syntax and providing a higher level of abstraction) and may therefore make the programming task easier even for professional programmers. This may be especially true during debugging, where graphics can be used to present much more information about the program state (such as current variables and data structures) than is possible with purely textual displays. This is one of the goals of Program Visualization. Other Program Visualization systems use graphics to help teach computer programming.