Advantages of a Visual Representation for Computer Programming

Kirsten N. Whitley (whitley@vuse.vanderbilt.edu)
Department of Computer Science, Box 1679 Station B
Nashville, TN 37235 USA

Laura R. Novick (Laura.Novick@vanderbilt.edu)
Department of Psychology, Box 512 Peabody
Nashville, TN 37203 USA

Doug Fisher (dfisher@vuse.vanderbilt.edu)
Department of Computer Science, Box 1679 Station B
Nashville, TN 37235 USA

Pictures and diagrams have long played an important role in human societies (Novick, 2001). Prehistoric peoples painted pictures on cave walls. The Bayeux Tapestry, from the 11th century, records the events surrounding the Battle of Hastings. In the 15th century, da Vinci made thousands of anatomical, mechanical, geographical, and other drawings. Today, diagrams are found on blackboards in most university departments (McKim, 1980). In our increasingly technical and technological society, diagrams (especially abstract ones) are likely to gain in importance. Even the traditionally text-driven field of computer programming now includes languages whose representations are diagrammatic rather than textual (Whitley, 1997).

LabVIEW™ (National Instruments, 1998), for example, features a programming language based on the dataflow paradigm in which the flow of information through the program is expressed using a notation that resembles circuit diagrams. LabVIEW™ was designed to facilitate the development of data acquisition, analysis, display, and control applications for science and engineering laboratories.

In an experiment with a 2 × 3 mixed-factors design, we assessed the comprehensibility of LabVIEW™'s representation. Representation type was manipulated between subjects, with 15 upper-level computer science students randomly assigned to receive LabVIEW™'s visual representation and 16 such students randomly assigned to receive an equivalent textual dataflow representation. The second factor – problem type (tracing, parallelism, debugging) – was manipulated within subjects.

The experiment involved a 90 min lecture, during which subjects were taught a subset of the LabVIEW™ language, followed by a 90 min test session. The test problems required subjects to read and understand code segments. For the three tracing problems, subjects were given input values for variables in the code and had to determine what the output values would be if the code were to execute. For the three parallelism problems, several program operators were highlighted, and subjects had to specify the order in which pairs of operators could execute. For the three debugging problems, subjects were given written specifications for the code and had to find the (single) error in the code.

For both solution accuracy and time, representation type interacted with problem type. For the more difficult parallelism and debugging problems, the visual representation was clearly superior to the textual representation: The visual subjects had higher accuracy scores and spent less time working on the problems. For the tracing problems, accuracy was similar for the two representations, but the visual subjects spent more time working on them. Overall, these results provide compelling evidence for the superiority of LabVIEW™'s visual representation over an equivalent textual representation.

The comprehensibility effects we found seem likely to generalize to novice LabVIEW™ programmers in more naturalistic situations. The subset of LabVIEW™ we used included a fair portion of the language. Moreover, our experimental tasks were representative of actual programming tasks and were sequenced in a natural instructional order. Although it is an open question whether the superiority of the visual representation will apply to other types of programming tasks (e.g., writing new code, modifying existing code), we suspect that it will under a variety of conditions.

References

McKim, R. H. (1980). *Thinking visually: A strategy manual for problem solving*. Belmont, CA: Wadsworth.

National Instruments (1998). LabVIEW™ User Manual [Software manual]. Austin, TX: Author.

Novick, L. R. (2001). Spatial diagrams: Key instruments in the toolbox for thought. In D. L. Medin (Ed.), *The psychology of learning and motivation* (Vol. 40, pp. 279-325). San Diego, CA: Academic Press.

Whitley, K. N. (1997). Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages and Computing, 8,* 109-142.