Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Int. J. Human-Computer Studies 64 (2006) 281–303

International Journal of Human-Computer Studies

www.elsevier.com/locate/ijhcs

Evidence in favor of visual representation for the dataflow paradigm: An experiment testing LabVIEW's comprehensibility

Kirsten N. Whitley[a], Laura R. Novick[b,*], Doug Fisher[a]

[a]Department of Computer Science, Vanderbilt University, Box 1679, Station B, Nashville, TN 37235, USA
[b]Department of Psychology and Human Development, Vanderbilt University, Peabody #512, 230 Appleton Place, Nashville, TN 37203-5721, USA

Received 14 December 2004; received in revised form 20 May 2005; accepted 29 June 2005
Available online 12 September 2005
Communicated by J. Scholtz

Abstract

This paper reports an experiment that examined the comprehensibility of the LabVIEW programming representation. LabVIEW is a visual programming language (VPL) based on the dataflow paradigm. The experiment compared LabVIEW to a semantically equivalent textual language using three types of tasks: (1) tracing problems: given code and input values, subjects were asked what output the code would produce if executed. (2) Parallelism problems: given code with several program statements highlighted, subjects were asked about the sequence in which those statements could execute. (3) Debugging problems: given code and its specifications, subjects were asked to find a logic error in the code. The experiment measured the subjects' time to solve the problems and accuracy of the answers. The subjects were upper-level university students who were intermediate-level programmers with no prior exposure to LabVIEW. Their performances showed differences due to representation for all three task types. Subjects using the textual representation completed the tracing problems significantly faster than subjects using the visual representation. In contrast, subjects using the visual representation were significantly faster for the parallelism problems and significantly more accurate for both the parallelism and debugging problems. These results contribute clear evidence for selected benefits of a visual representation for small-sized code segments, evidence that LabVIEW succeeds in highlighting data dependencies, and evidence that LabVIEW helps programmers to maintain an overview of their code. The authors consider the implications of these results for VPLs more generally.

Keywords: Visual programming; LabVIEW; Empirical evaluation

1. Introduction

Visual programming research rests on the idea that visual representations can be properly integrated with text to form visual programming languages (VPLs) and visualization systems that will improve human programming experience. The anticipated improvements are hypothesized to arise because the visual representations bring programming closer to the human side of the human–machine interface, just as high-level languages tipped the accessibility scales relative to assembly languages. With such a design tenet, an important question is, "What empirical data exist to show when and how visual representations can be beneficially used in programming?" The past 20 years have witnessed the emergence of an active visual programming research community whose efforts have produced many VPLs and visualization systems. However, there has been a relative paucity of empirical studies addressing the types of programming tasks for which VPLs are useful. The relative lack of evidence supporting visual programming representations is one of the bigger open problems for the visual programming research community.

This paper reports a controlled experiment designed to explore the cognitive effects of the visual representation in LabVIEW for intermediate-level programmers with no prior exposure to this language. LabVIEW, which stands for Laboratory Virtual Instrument Engineering

*Corresponding author. Tel.: +1 615 343 6072; fax: +1 615 343 9494.
E-mail addresses: Laura.Novick@vanderbilt.edu (L.R. Novick), dfisher@vuse.vanderbilt.edu (D. Fisher).

Workbench, is a VPL and environment. Commercially available since 1986, LabVIEW is one of the more widely used VPLs to date. Santori (1990), Kodosky et al. (1991) and Vose and Williams (1986) give early descriptions of the language; Johnson (1997), Wells and Travis (1997) and Wells (1995) supply more thorough descriptions. The LabVIEW programming language is based on the dataflow paradigm and is designed to facilitate development of data acquisition, analysis, display and control applications. These tasks, which are common in science and engineering laboratories, comprise the application area known as instrumentation. Although somewhat specialized in its usage compared to a language such as C, the applicability of LabVIEW is considerably more general than that of a graphical language such as relay ladder logic, which has long been used for configuring programmable logic controllers.

LabVIEW is interesting for empirical investigation because (a) it is one of the few general-purpose VPLs (Kiper et al., 1997), (b) it is widely used and (c) prior studies evaluating its visual representation have yielded seemingly conflicting results. Rather than testing hypotheses about individual LabVIEW constructs, our study is more holistic in nature, examining the potential benefits of LabVIEW's integrated visual representation compared with that of a semantically equivalent textual language. Because our experiment considers LabVIEW's visual representation as a whole, we are unable to determine which aspects of the representation are responsible for which aspects of our results. Nonetheless, our findings suggest pros and cons of basic programming constructs for conditional logic and iteration that can be tested in subsequent studies. More generally, the experiment contributes to the growing research about the impact of VPLs (Whitley, 1997), suggesting visual constructs that aid comprehension and manipulation of programs in various contexts.

We begin with a general survey of the literature, highlighting the task-specific nature of advantages of differing representation schemes. This is followed by a description of LabVIEW and the equivalent textual language devised for this experiment. Next, a more specific literature review summarizes previous studies of LabVIEW, which includes an opinion survey of professional LabVIEW programmers. The survey's results were helpful in guiding the creation of program segments in the current experiment. Subsequent sections present the experiment's methods, results and implications.

2. Literature review

An underlying premise of visual programming research is that visual representations can improve the human–machine interface by catering to human cognitive abilities. For example, visual representations capitalize on the strengths of the human perceptual system by substituting easier perceptual inferences for more computationally intensive search processes and sentential deductive inferences (Larkin and Simon, 1987; Barwise and Etchemendy, 1991). This is not to say, however, that visual representations are always superior. Rather, the advantage of one representation over another depends on the fit of the representation's organization of information to the structure of the environment and the cognitive demands of a task, as well as on people's prior knowledge and experience (e.g., McGuinness, 1986; Sanfey and Hastie, 1998; Neary and Woodward, 2002).

2.1. *Representational advantage varies with task and problem size/complexity*

Green (1977) and Gilmore and Green (1984) proposed, and provided support for, what they termed the *match–mismatch hypothesis*, which specifies that every representation highlights some kinds of information, while obscuring other kinds. Thus, it follows that every representation will likely perform well for some tasks and poorly for others. For example, Green and Petre (1996) experimentally confirmed that one form of conditional logic available in the VPL, LabVIEW, facilitated backward reasoning and the other facilitated forward reasoning. However, their study also found that response times on a series of comprehension questions were longer (i.e., worse) using LabVIEW's visual constructs than with corresponding textual representations.

The match–mismatch principle is also observable in McGuinness' (1986) study of a tree versus a matrix representation to encode information about family relationships. The matrix used in Experiment 1 led to better performance than did the tree on questions about appropriate vacation partners, but representation had no significant effect on questions concerning family inheritance. In Experiment 2, the family relationship information was mapped onto the two representations in a "reversed" manner, leading to a reversal in the pattern of results: The matrix representation led to better performance for the inheritance questions, but there was no difference for the vacation partner questions.

Chattratichart and Kuljis (2002) investigated diagrammatic representations of control flow and dataflow. Novice programming subjects were tested on various representations of each of these two paradigms, as well as on an equivalent textual program. They found clear advantages for certain representations over others in terms of response time and accuracy of answers on test questions. All the diagrammatic representations were motivated by constructs found in the literature, most of which led to better performance than text, but there were notable exceptions.

An observational study of spreadsheets by Hendry and Green (1994) underscores the subtle nature of the match–mismatch hypothesis. The marketplace success of the spreadsheet makes it clear that the spreadsheet is useful; nevertheless, the spreadsheet does not facilitate all