Testing the Usability of Visual Languages: A Web-Based Methodology

Mauro Mosconi – Marco Porta

Dipartimento di Informatica e Sistemistica – Università di Pavia
Via Ferrata, 1 – 27100 – Pavia – Italy
mauro@vision.unipv.it – porta@vision.unipv.it

1 The Objective: Testing the Expressiveness of Visual Control Structures

The purpose of this paper is to illustrate the methodology we developed in order to start a comparative usability study for different implementations of visual control flow constructs.

Since loops in data-flow visual languages (Hils 1992) may be difficult to understand (Green and Petre 1996), we decided to test the usability of the solutions we devised for *VIPERS*, a data-flow visual language developed at the University of Pavia (Ghittori, Mosconi and Porta 1998). To get useful indications, we opted for a comparative analysis, also referring to the well-known data-flow language *LabView* (Vose 1986), where iterative constructs are implemented according to a totally different philosophy.

Among the possible evaluation methods (Preece 1993), we elicited *observational evaluation*, which involves observing or monitoring users' behavior while they are using an interface, and *survey evaluation*, which means seeking users' subjective opinions. To collect data about what users do when they interact with the test interface, employment of *direct observation* was avoided. In fact, if users are constantly aware that their performance is being monitored, their behavior may be strongly influenced (*Hawthorne effect*). Instead, we used *software logging* to record the dialog between user and system. In particular, our methodology is based on the use of the *log files* of a web server, as will be illustrated. Moreover, we elicited *questionnaire* forms to support the survey evaluation.

Published in the Proceedings of the 8th International Conference on Human-Computer Interaction (HCI'99), Vol. 1, 22-27 August 1999, Munich, Germany.

2 The Testing Methodology

2.1 Selecting the Test Context

Every usability evaluation is meaningful within a precise context, including the practice level of testers, the types of task being undertaken and the environment in which the test is carried out.

For our first experiments, we decided to work with high school students (17 to 19 years old) with little skill in textual programming and no experience at all in visual programming. We set proper mathematical applications as test tasks. Even though we were aware that many other application domains would be more suitable for a data-flow approach, we opted for problems close to their school experience.

As far as the programming environment was concerned, our aim was to make the interaction independent of the computer platforms used (by carrying out tests in an heterogeneous environment –our lab– with both PCs and Mac and UNIX machines). This last consideration also influenced our choice to focus on the program understanding process rather than on program construction.

We stress the fact that we did not want to compare the usability of the whole VIPERS and LabView environments, but only to observe how efficiently these two languages visually express control constructs (loops, in particular).

2.2 Planning the Tests

We planned two sessions, each one with twelve users. Altogether, twelve users tackled a set of three problems through VIPERS and twelve the same set through LabView.

Each user had to examine, in sequence, three visual programs displayed on the computer screen and translate them into correspondent textual programs (in pseudo-Pascal, since they all knew this language). We considered the number of right solutions as a first indicator of the comprehensibility of the languages in the loop implementation. Time for the test was set at one and a half hours.

2.3 Implementing the Tests: the Technical Approach

The idea making the creation of our tests (and generally any tests that can exploit this set up) particularly economical is the use of a web server and its log files. Each problem is presented to the user in the form of a web page: during the test, the user interacts solely and exclusively with a web browser, independently of the platform used. In the web page (see Figure 1) the problem is visualized as an image, or rather, as an *image map*. Image maps are graphics

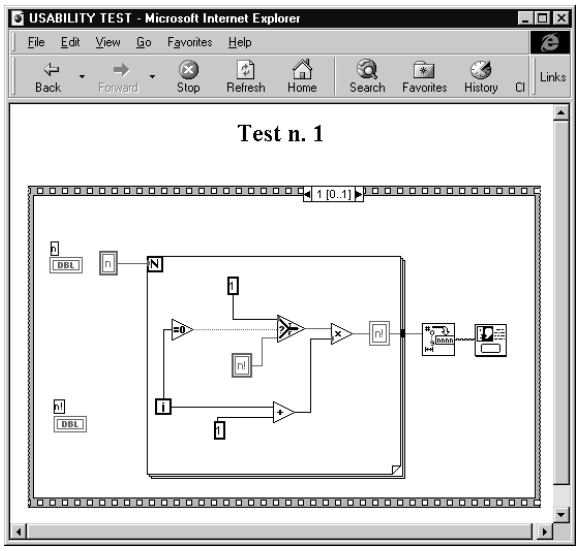

Figure 1: a (LabView) visual program to be examined is shown within a web browser

where certain regions are mapped to URLs. By clicking on different regions, different resources can be accessed from the same graphic.

In our case, we associated each graphic symbol with a page illustrating its meaning. The log file of the web server, by registering the actions of the users, was able to reveal how many times (and in what order) each user had clicked on a symbol of the program to ask for the help pages, and how much time the user stopped to read the explanations before returning to the main page using the BACK key.

Figure 2 points out, using “fingers marks”, those parts of one of the test programs that were the most studied by an average of testers. For a complete report of the results obtained see (Ghittori 1998): here, we concentrate on illustrating the methodology used.

One type of result we believe worth showing is that given in Figure 3, where the path followed by a tester in examining a program was reconstructed on the basis of the log file. Our impression is that this scan path (which can easily be obtained automatically) may prove to be a very interesting tool (as analyzed by cognitive psychologists) with which to verify old hypotheses regarding the mental representation of visual programs and the cognitive processes involved in programming, in these new environments.

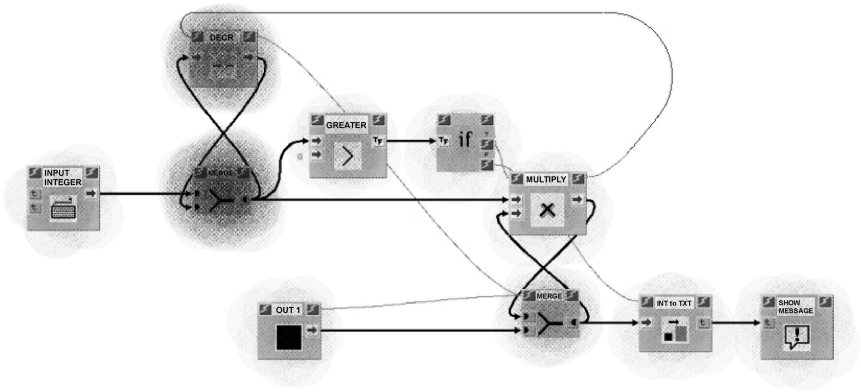

Figure 2: a finger-marks representation of the mostly accessed elements in a (VIPERS) program understanding process (derived from the web-server log file)

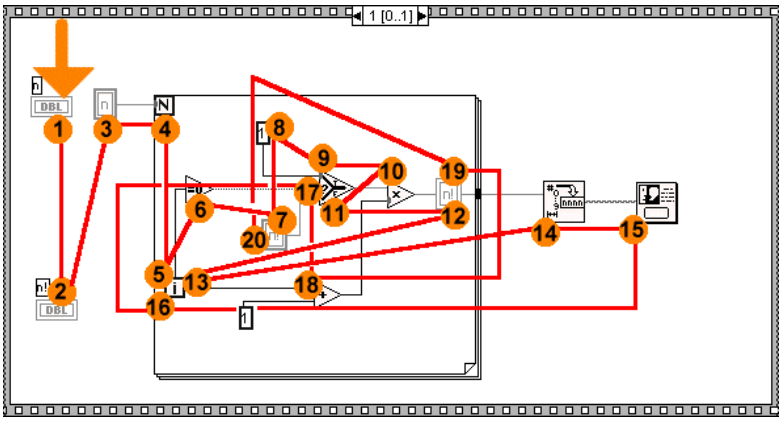

Figure 3: reconstruction of the mental scan-path of the user inspecting a visual program

3 Considerations About the Testing Methodology

We believe we have perfected a methodology for the usability analysis which is effective, supplies a wealth of feedback, and is moreover fast and economical. In particular, we would like to point out that:

- the method proposed is non-invasive (the user does not know he/she is being monitored and is therefore less subject to external conditioning).
- Data collection is objective because it is given by the log file path.

- This method can be used on different hardware platforms at the same time and allows more than one user to be monitored at the same time.
- The tests can be carried out without having to remove the tester from his/her work.
- The processing of the data gathered is made easier by numerous elaboration tools for the web log file information. Moreover, setting the test does not require particularly extensive knowledge of informatics.
- The hypermedial means used make it easier to prepare the material needed to instruct the testers (in our case, these are the pages explaining how the visual language "blocks" function). In other words, the problem is solved of how to organize the presentation of information to the user by allowing him/her to directly access only the information needed (which varies according to the ability of the user and the type of problem).
- By registering the position of the click on the image it is also possible to automatically obtain a graphic representation of the "mental path" taken by the subject analyzing a visual program.
- In general, it makes sense to state that this methodology can be applied to a variety of different research environments where interaction between users and complex images is being studied.

References

Ghittori, E. (1998). *Usabilità dei linguaggi visuali dataflow: il problema dei costrutti di controllo*. Master's Thesis, University of Pavia.

Ghittori, E., Mosconi, M., Porta, M. (1998). Designing new Programming Constructs in a Data Flow VL. *Proceedings of the 14th IEEE International Conference on Visual Languages* (VL'98, 1-4 September 1998, Nova Scotia, Canada).

Green, T. R. G., Petre, M. (1996). Usability Analysis of Visual Programming Environments: A 'Cognitive Dimension' Framework. *Journal of Visual Languages and Computing*, 7(2), 131-174.

Hils, D. D. (1992). Visual Languages and Computing Survey: Data Flow Visual Programming Languages. *Journal of Visual Languages and Computing*, vol. 3, 69-101.

Preece, J. (1993). *A Guide to Usability. Human Factors in Computing*. Addison Wesley.

Vose, G. M. (1986). LabView: Laboratory Virtual Instrument Engineering Workbench. *BYTE*, vol. 11, n. 9, 82-84.