WIPE – Pilot Testing and Comparative Evaluation

Vassilios Efopoulos
Department of Applied Informatics, University of Macedonia,
efop@uom.gr

Georgios Evangelidis
Department of Applied Informatics, University of Macedonia,
gevan@uom.gr

Vassilios Dagdilelis
Department of Educational and Social Policy, University of Macedonia,
dagdil@uom.gr

Abstract

This paper discusses the pilot testing and evaluation of a database-driven web-based programming environment called WIPE (Web Integrated Programming Environment). WIPE is a teaching tool for secondary education students that are introduced to the principles of programming. The programming environment was used in secondary schools in Greece and the results of its evaluation demonstrate that it successfully deals with the difficulties novices meet.

1. Introduction

Programming constitutes a difficult learning process ([5], [2]). The term "difficult" is used to imply, that, novice programmers usually cannot comprehend even small programs or provide code for solving simple problems [3]. In general, this is attributed to the fact that programming requires novices to [4]:

1) Devise procedures that solve generic classes of problems (this is the essence of algorithmic solutions),
2) Adapt these procedures in an abstract solution model (i.e., implement an algorithm), and
3) Code them in a concrete way (use a programming language).

Moreover, a novice programmer has to understand how to use the graphical user interface of the programming environment in use.

A great number of programming environments and languages have been developed in order to introduce novices to the principles of programming, like Thetis [7], DrJava [1], BlueJ [8], JJ [9].

However, most of these environments and languages have been negatively criticized for being too technocratic and for not adhering to the recent findings in pedagogy and didactics [12], despite their pedagogical nature. Thus, for example, in most cases the error messages produced by such programming environments are cryptic for the novice (non-professional) programmer.

WIPE was designed taking into consideration the latest findings on the difficulties novice programmers meet and the proper ways to address them. WIPE constitutes an attempt to incorporate the experience and knowledge that has been accumulated by the related research in the subject of teaching of programming. WIPE aims at introducing secondary education students to the principles of programming. The programming environment is attractive and easily accessible via a web browser and the programming language used is simple, minimal and quite similar to Pascal. The programming environment incorporates a lot of features (GUI, multiple representations, step-by-step execution of source and assembly code, error messages in natural language) that aim at helping novice programmers. WIPE is supported by a DBMS and exploits the possibilities offered by a web-based application.

In Section 2, we present the design principles behind WIPE. The interested reader can find a more detailed discussion in [6]. Section 3 states the hypotheses we wanted to test and Section 4 describes the data acquisition method we followed. In Section 5, we present the results of the comparative evaluation of WIPE and a commercial programming environment. We conclude in Section 6 with a general presentation of the findings of the evaluation of WIPE.

2. Design Principles

The design of WIPE is characterized by simplicity and consistency. WIPE is addressing the needs of novice programmers. A novice can easily learn to use the programming environment. This objective is achieved thanks to an ergonomic and functional GUI, without the need of installation (since it is accessible

via the web). The programming language that is incorporated in WIPE can be seen as a subset of Pascal; it is very simple and it can be used as an introductory programming language.

The consistency in WIPE is achieved by a strict definition of the syntactically correct programming constructs and their semantics. Thus, for example, the syntax of I/O commands is identical for all data types and each construct has unique syntax.

WIPE does not incorporate tools for automatic generation of code or for implementing a graphical user interface. Such tools are usually oriented towards advanced programmers and application developers. The approach of WIPE helps inexperienced users become quickly familiar with the programming environment and write source code, without limiting their imagination and expressiveness.

The programming language of WIPE has the following features:

1) *Web-based operation*. The compiler operates via a web browser. Moreover, students can follow all the intermediate stages of the construction of a program via a web browser, i.e., the compilation, the resulting assembly code, the intermediate values of the virtual machine registers and program variables.
2) *Support of a symbolic language (Assembly)*. During the process of compilation the source code is translated to an equivalent symbolic language code (assembly language). The symbolic language produced by the WIPE Compiler is a pseudo-assembly that is executed in a virtual machine with two registers.
3) *Systematic recording of students' actions (Recordability)*. The actions of students (compilations, program executions with the associated input and output values) are stored in a database and are accessible for evaluation by the teachers.
4) *Comprehensible error messages*. The error messages generated by the WIPE Compiler are in natural language and are suitable for assisting novices debug their code.

3. Hypotheses

We decided to pilot-test WIPE and evaluate it in a school setting in order to get feedback from students and teachers and also get answers on some important questions (hypotheses):

1) *Attractiveness.* Does the programming environment as a whole aid the learning process? Do students easily adopt it? Are they interested in it? Have they used it effectively? How do teachers and students evaluate its various features? Do teachers believe that it is a useful tool for teaching programming? We consider that the provocation of user interest is a very important factor for a successful evaluation procedure and in general for the adoption of new software. Our hypothesis is that our programming environment is equipped with all the appropriate features that make it attractive to students and teachers.
2) *Error handling.* Is there a relation between the usage of the programming environment and the errors novice students make when they are introduced to programming? Do students make less or more errors when using WIPE compared to other programming environments? Does the programming environment produce clear and easy to understand error messages? We claim that knowledge acquisition is faster and easier when the programming environment aids students in minimizing their errors.
3) *Usability.* Are there certain characteristics of the programming environment that students find helpful and others that stand in their way? Is the environment easy to use? Should some components of the software be redesigned? Is there a need for additional features that are missing? It has been shown [10] that its indented users better evaluate the usability of a programming environment through testing in every day conditions. During the design of WIPE we followed the usability guidelines of Nielsen [11]. Our goal was to help its users learn to use and operate it as easily as possible. Despite that, only thorough testing in a school laboratory and the evaluation of the remarks of students and teachers can prove whether our design is successful or not.

4. Data Acquisition

The results of the evaluation procedure were based upon the following data:

1) Direct observation of the way students worked during the courses and recording of the difficulties they met in understanding the taught concepts and in using the programming environment.
2) Analysis of the successive versions of the student programs that were recorded in the database via the recordability feature of the programming environment. That is, anytime a student compiled or ran a program the system recorded all the relevant information.
3) Recording, comparison and analysis of the programs that were written by students that participated in the comparative evaluation of WIPE and a commercial environment used in Greek schools (Borland Turbo Pascal is the standard

programming environment used in secondary education in Greece).

4) Answers to the questionnaire of the pilot run by the participating students. That questionnaire was given after the completion of the test course.
5) Interviews with the teachers that were responsible for the evaluation procedure.

The recording procedure resulted in the collection of thousands lines of source code since every record in the database was the source code of a compiled program. Of course, it was the case that many successive programs recorded in the database were very or quite similar. This is because students were compiling very often, even when they were making minor changes to the source code. The analysis of the programs and the identification of the successive versions were made possible via the usage of the teacher's tool.

It was quite easy to analyze the syntax errors that were detected thanks to the mechanism for recording the students' actions. The logic errors were detected manually by examining the source code of each program.

The observation of the students while working with WIPE and the analysis of the data collected in the database revealed that certain students used WIPE to experiment and they wrote code for programs not included in the suite of the tutorial exercises. Also, many students used WIPE via their home computers as indicated by the timestamps recorded for their compilations and program executions.

5. Comparative evaluation

The comparative evaluation between WIPE and Borland Pascal took place in November and December of 2004. The participants were 28 students of the 11th grade of the 6th Technical/Vocational School of Thessaloniki that were divided into two groups of 14 individuals each. All students had been recently (September 2004) introduced to programming with the use of the programming environment of Borland Pascal for Windows. The students of the first group attended a 4-hour seminar to become familiar to WIPE and its programming language. The first group used WIPE (group A) whereas the second group used the already familiar environment of Borland Pascal (group B).

The same suite of programming exercises was administered to each group. The exercises were designed to be as simple as possible and included trivial programs. They required fairly short programming solutions but also encouraged the use of programming constructs like conditional statements, loops, arrays and functions.

The process of recording successive solutions in Borland Pascal was achieved by following the convention of saving the current state of the program before each compilation. Each version was assigned a new serial number (program1.1, program1.2, etc.).

In WIPE the intermediate results, i.e., the source code for each successive compilation and the output of each execution, are stored in a MySQL database, thus, it was easy for us to collect the necessary information.

In the analysis of the source code there were two major categories of errors: syntax and logic errors. The results from the analysis of students' errors are shown in the tables below:

Table 1. Syntax errors per student

Syntax errors (per student)	WIPE (group A)	Borland Pascal (group B)
Mean	9,79	17,21
Standard deviation	4,35	5,35
Number of students	14	14

Table 2. Logic errors per student

Logic errors (per student)	WIPE (group A)	Borland Pascal (group B)
Mean	3,29	5,21
Standard deviation	1,38	2,08
Number of students	14	14

We observe that the average number of errors in WIPE is substantially lower than in Borland Pascal for both syntax and logic errors.

The distribution of error frequencies shown in Figure 1 and in Figure 2, are relatively smooth, if we take into consideration the small number of the sample (14 students). We observe that for both syntax and logic errors the distributions for WIPE are obviously different from the corresponding ones for Borland Pascal and they reveal that students make more errors with Borland Pascal than with WIPE. The students who used WIPE demonstrate a considerably smaller frequency of errors compared to the students who used Borland Pascal. The mean of syntax errors for the group of WIPE students was 9,79 with a standard deviation of 4,35 while for the group of Borland Pascal students the mean was 17,21 with a standard deviation of 5,35.

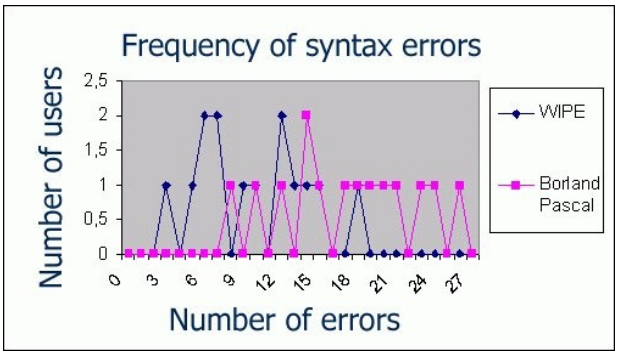

Figure 1. Frequency of syntax errors

In the case of logic errors we had similar results with the WIPE group having 3,29 errors on average with a standard deviation of 1,38 and the Borland Pascal group having 5,21 errors on average with a standard deviation of 2,08.

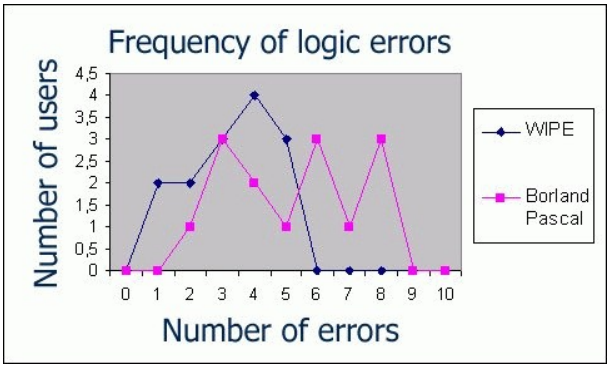

Figure 2. Frequency of logic errors

We decided to use statistical methods (hypotheses tests) in order to compare the means so that we further support our initial hypothesis, which was that: "The choice of a suitable programming environment has a direct impact in the type and the frequency of errors that novice programmers make".

An independent t-test shows that the two groups exhibit statistically significant differences in the means for both syntax errors ($t = 4,029$, $df = 26$, $p = 0,000433 < 0,01$) and logic errors ($t = 2,887$, $df = 26$, $p = 0.007728 < 0,01$).

Regarding the choice of the t-test for the comparison of the means between the two (independent) samples, we performed the appropriate tests and the outcome was that t-test was suitable. The process was the following:

(a) *Normality tests of the two samples.* We selected the well-known Kolmogorov-Smirnov test and the Shapiro-Wilk normality test which is used when the size of the samples is smaller than 50. The results of these tests are presented in Table 3 (where df are the degrees of freedom). According to the levels of importance (sig.) that were calculated, we can claim that the samples follow the normal distribution ($p > 0,05$).

(b) *Comparison of the variances of the two samples with the F-test.* This test leads to the conclusion that the variances are equal (non-significantly different) for both syntax errors (F-test = 0,556 and $p = 0,463 > 0,05$) and logic errors (F-test = 4,113 and $p = 0,053 > 0,05$).

Table 3. Normality tests of the two samples

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	GROUP	Statistic	df	Sig.	Statistic	df	Sig.
LOGICAL	A	,089	14	,200*	,984	14	,978
	B	,167	14	,200*	,968	14	,802
SYNTAX	A	,149	14	,200*	,918	14	,275
	B	,197	14	,145	,909	14	,208

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

The above results confirm that the choice of the t-test is suitable for the comparison of the means of the two samples.

6. Conclusions

The pilot testing of WIPE and the analysis of the evaluation results reveals that our initial goals were satisfied. We briefly present them below.

1) *Attractiveness*. The proposed programming environment and the accompanying suite of tutorial exercises in general attracted the interest of students and helped them comprehend the basic principles of programming. This conclusion results from the student answers in the evaluation questionnaire and the analysis of the information recorded in the WIPE database. The majority of students (80%) consider that the tutorial exercises that accompany WIPE helped them to better comprehend the principles of programming. The teachers that taught programming using WIPE, initially they were quite cautious but day after day they started making positive comments, expressing their satisfaction for taking part in the pilot testing of WIPE. The teacher tool was a welcome surprise for the teachers, especially when they realized the wealth of information it can provide them with.

2) *Error handling*. The compiler messages were simple and comprehensible to the average student. Most students consider that the messages are comprehensible and facilitate the pinpointing of errors (average 4 in a 1-5 scale), while 86% of students classify the error messages as one of the features of the programming environment that most satisfied them. The group of students that used WIPE made less logic errors than the one that used Borland Pascal. Of course, no programming environment can eliminate the errors novice students make. However, WIPE assisted students in easily correcting their syntax errors by providing comprehensible error messages, colorized keywords, step-by-step execution and variable watch. The recorded information on the WIPE database reveals that students took great advantage of the step-by-step execution feature (45% of all program executions were step-by-step executions). This feature satisfied 83% of students while 94% of them evaluated positively the feature of watching the intermediate values of program variables and system registers. Students also tend to make less syntax errors when using WIPE instead of Borland Pascal.
3) *Usability*. The students were eager to explore the programming environment and met no problems in using it. WIPE was proved user-friendly and functional. Most of the students wished they could use WIPE when studying at home (81%) and their textbooks are accompanied by similar (easy to use) software (78%).

Our final conclusion was that WIPE constitutes a useful and powerful tool in the hands of teachers when teaching the principles of programming to novices.

A demo version of WIPE is accessible at http://eos.uom.gr/~efop.

7. References

[1] Allen, E., Cartwright, R., Stoler B. (2002) "DrJava: A Lightweight Pedagogic Environment for Java", SIGCSE 2002.

[2] Bonar, J., Soloway, E. (1985). "Preprogramming Knowledge: A Major Source of Misconceptions in Novice Programmers". In Human-Computer Interaction, 1(2):133-161.

[3] Bruckman, A. & Edwards, E. (1999). "Should We Leverage Natural-Language Knowledge? An Analysis of User Errors in a Natural-Language-Style Programming Language". In Proceedings Computer Human Interaction 1999 (CHI'99), Pittsburgh, USA, May.

[4] Dagdilelis V. (1986). "Conceptions des eleves a propos des notions fontamentales de la programmation informatique en classe de Troisieme", Memoire D.E.A., Universite Joseph FOURIER, Grenoble, France.

[5] Du Boulay, B. (1989). "Some Difficulties of Learning to Program". Studying the Novice Programmer. E. Soloway and J. C. Spohrer. Hillsdale, NJ, Lawrence Erlbaum Associates: 283-299.

[6] Efopoulos, V., Dagdilelis, V., Evangelidis, G., and Satratzemi, M. (2005). "WIPE: A programming environment for novices. Design Principles and Evaluation", In Proceedings ACM ITiCSE 2005 (to appear).

[7] Freund, S., and Roberts, E. (1996). "THETIS: An ANSI C Programming Environment for Introductory Use", SIGCSE Bulletin, February, Vol. 28(1), 300-304.

[8] Kölling, M. (1999). "Teaching Object Orientation with the Blue Environment". Journal of Object Oriented Programming, 12(2), May, pp14-23.

[9] Motil, J., Epstein, D. (2000). "JJ: a Language Designed for Beginners (Less Is More)", http://www.ecs.csun.edu/~jmotil/TeachingWithJJ.pdf.

[10] Nielsen, J. (1993). "Usability Engineering". Academic Press.

[11] Nielsen, J., and Molich, R. (1990). "Heuristic evaluation of user interfaces", Proc. ACM CHI'90 Conf. (Seattle, WA, 1-5 April), 249-256.

[12] Pane, J. & Myers, B. (2000) *The Influence of the Psychology of Programming on a Language Design: Project Status Report*. 12th Annual Workshop of the Psychology of Programming Interest Group, Corigliano Calabro, Italy.