WIPE: A Programming Environment for Novices

Vassilios Efopoulos
Department of Applied Informatics
University of Macedonia
156 Egnatia Str.
Thessaloniki, Greece
+30-2310-889155
efop@uom.gr

Vassilios Dagdilelis
Department of Educational and Social Policy,
University of Macedonia
156 Egnatia Str.
Thessaloniki, Greece
+30-2310-891336
dagdil@uom.gr

Georgios Evangelidis
Department of Applied Informatics
University of Macedonia
156 Egnatia Str.
Thessaloniki, Greece
+30-2310-891844
gevan@uom.gr

Maya Satratzemi
Department of Applied Informatics
University of Macedonia
156 Egnatia Str.
Thessaloniki, Greece
+30-2310-891897
maya@uom.gr

ABSTRACT

This paper presents an overview of the design principles and the evaluation of a new programming environment, WIPE (Web Integrated Programming Environment), designed specifically to teach novices the fundamentals of programming. The environment is designed for use in secondary education as a first programming course, in order to help students become familiar with the main programming concepts.

Categories and Subject Descriptors

K.3.1 [**Computer Uses in Education**]: *Computer-assisted instruction (CAI),* K.3.2 [**Computer and Information Science Education**]: *Computer science education,*

General Terms

Algorithms, Design, Languages

Keywords

Web-based compiler, e-learning, interactive learning environment, programming and programming languages, secondary education

1. INTRODUCTION

Research in the area of Computer Science Education has been quite active during the past 25 years. The main focus of the research was on the difficulties novice programmers meet when they attempt to program a computer. Programming appears to be the hardest faculty to master when dealing with computers. Since the seminal work of Weinberg [22], hundreds of related papers have been published ([5], [6], [19], [12], [11], [9], [3], and [4]). In addition, during all these years we had the emergence of journals dedicated to the teaching of Computer Science (e.g., *Computer Science Education* [7]), or journals that regularly published papers about psychology and the teaching of programming (e.g., *International Journal of Human-Computer Studies,* formerly *International Journal of Man-Machine Studies* [14]).

Annual symposia and conferences that deal exclusively with Computer Science Education have been established (e.g., *ITiCSE/SIGSCE* [2], and *NECC* [16]), and, in the past, a series of workshops under the general title *Empirical Studies of Programmers* were organized. Well-established organizations are keenly interested in the teaching of Computer Science; in addition to the likes of ACM and IEEE that are closely related to the science and profession of Computer Science, organizations like AACE [1] and ISTE [16] systematically deal with the teaching of Computer Science, Informatics and, especially, Computer Programming. We should also mention the existence of special interest groups (e.g., PPIG [21]), and groups of researchers in universities and research centers.

Despite the great interest in the teaching of Computer Science and the relatively high number of research studies, we claim that a *general theoretical framework* is evidently still missing. This is not the case in Mathematics and Physics, two other scientific areas that deal with similar subjects for a longer period of time compared to Computer Science. The lack of a theoretical framework in the case of Computer Science and more specifically Programming, unfortunately, in effect promotes an empirical way of tackling issues that does not allow the integration of the vast data and findings reported in the research literature, does not distinguish the most relevant research methods, and, does not permit the methodical verification of the reported ascertainment. As a result, one gets the feeling that simple findings, isolated facts, or even curriculums or new languages and tools ([17], [18]) are being designed and redesigned in an attempt to overcome the difficulties novice programmers meet without referring to a general framework that would allow for a possible unification of the findings and also provide a means to verify them. The validity of such findings is, thus, questionable.

In this paper we present the design principles of WIPE (Web Integrated Programming Environment). WIPE is an educational software we have developed to introduce novices to programming.

We also report on the feedback we collected by extensively using WIPE in secondary education schools. WIPE is based on a previous project that did not focus exclusively on the teaching of Programming ([10], [13]). The design of WIPE was based upon and was influenced by some fundamental didactic principles and the experience obtained by former research regarding the teaching of introductory concepts on Programming. WIPE is also equipped with some powerful tools targeting the teacher, rather than the student. These tools can assist teachers pinpoint the specific areas where students have difficulties. Finally, to make its use more effective, WIPE is accompanied by educational material (exercises and *didactic situations*).

In Section 2 we present the design principles of WIPE and in Section 3 its main features. The results of an evaluation of WIPE in three secondary education schools are presented in Section 4.

2. DESIGN PRINCIPLES

Although the OOP model dominates the international teaching practice, the greek secondary education system still uses the model of imperative programming. Thus, WIPE supports the teaching of imperative programming. Thanks to its modular design, though, a transition to an OO language is a quite straightforward task to accomplish.

In the following subsections we address the basic design principles WIPE follows.

2.1 Simplicity

The graphical user interface of WIPE is user friendly and easy to operate. WIPE requires no installation - it is accessible via the Internet or the school intranet and all that is needed to use it is a web browser. One of our primary design goals was to offer novice programmers an easy-to-use programming environment. It is a well-documented fact that most programming environments feature a quite complex user interface [6].

These environments are, primarily, designed for professional programmers and the complexity of their tools renders their use prohibitive by novice programmers. Compared to commercial environments, WIPE includes a small set of tools to aid novices understand the basic principles of programming and the way a program is executed, rather than increase their programming productivity. Students can start using the programming environment of WIPE without any formal introduction. After the first half hour of their first session with WIPE, they can use it effectively. Thus, we have designed a programming environment that cannot be exploited by professional programmers, because it is oriented to novice programmers and their learning needs.

2.2 Consistency

The need and importance of consistency in a programming language and a programming environment is a subject that has been extensively addressed by the research community, quite often with dissenting opinions, especially when regarding the language syntax issue ([19], [8]). WIPE attempts to avoid all kinds of inconsistencies. For example, the syntax of I/O commands remains identical for all data types. The differentiation that one can observe in languages like C (for example in the syntax of scanf and printf) undeniably offers greater flexibility to the programmer but it, almost always, confuses novice programmers. In addition, each language statement has a unique syntax and all its constituent elements have unique semantics. For example, subroutine parameters are passed by value (and never by reference) and there is a single way of updating the values of variables and table elements (in contrast to C/C++ that support multiple alternative ways of doing things).

This limitation in the programming capabilities of the language supported by WIPE is, in our opinion, consistent to a fundamental teaching principle, according to which complex concepts are usually taught via successive approaches in a school setting (a spiral like approach). For example, many complex concepts in Mathematics and Physics are taught again and again in different grades, each time at an increasing degree of difficulty/complexity. Similarly, in WIPE the initial approach of programming concepts is addressing students that should become familiar with these concepts at an elementary level. These students are not professional programmers and they should not become familiar with the full range of programming concepts and programming tools.

2.3 Emphasis on the source code

In accordance to the above rationale, WIPE emphasizes on the production of source code and it does not come with code generation tools or graphical user interface design aids. Our goal is to have the student concentrate on writing source code and not on designing a user interface. Such an approach helps novice programmers get acquainted quickly to the programming environment and produce source code, without inhibiting their imagination and expressiveness.

Also, the time students spend in learning the programming language and its syntax should not hinder their learning of the fundamental programming and algorithmic concepts. WIPE attempts to adhere to its main goal that is the teaching of programming. Students should spend more time learning than debugging. Otherwise, they can become discouraged since they spend most of their time dealing with secondary (organizational) issues and not the problem at hand.

3. FEATURES

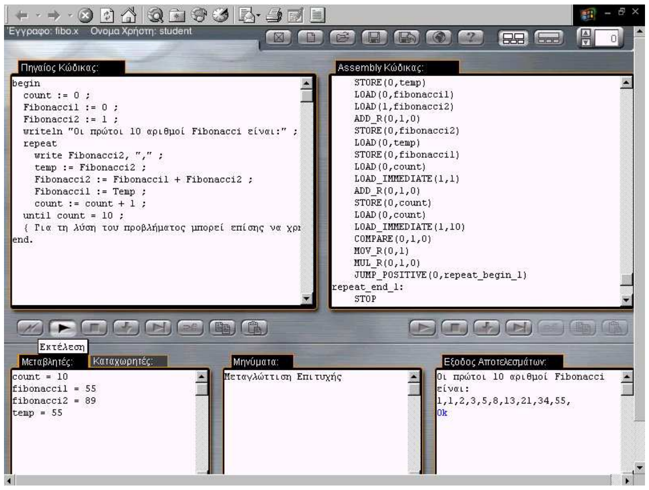
The programming environment of WIPE sports certain features that are dictated by its indented didactic use.

3.1 Web-based operation

The whole programming environment is web-based. Students can use it in any computer that is connected to the school intranet or the Internet and is equipped with a web browser.

3.2 Visualization tools

WIPE Compiler is the most important tool of WIPE. It is a specially designed web-based compiler implemented in Macromedia Director (Shockwave output), that, in its current version, compiles the source code of a simple, imperative, Pascal-like programming language to a pseudo-assembly code. WIPE Compiler gives students the freedom to experiment by allowing them to write their programs, check their correctness, visualize their execution by observing the intermediate values of the machine registers and the program and temporary (compiler) variables and observe their output. Figure 1 displays a snapshot of the WIPE Compiler after a successful compilation of a source program (top left window), the corresponding variable watch window (bottom left), the messages window (bottom center) and the output window (bottom right).



```
begin
  count := 0 ;
  Fibonacci1 := 0 ;
  Fibonacci2 := 1 ;
  writeln "Οι πρώτοι 10 αριθμοί Fibonacci είναι:" ;
  repeat
    write Fibonacci2, "," ;
    temp := Fibonacci2 ;
    Fibonacci2 := Fibonacci1 + Fibonacci2 ;
    Fibonacci1 := Temp ;
    count := count + 1 ;
  until count = 10 ;
  { Για τη λύση του προβλήματος μπορεί επίσης να χρ
end.
```

```
    STORE(0,temp)
    LOAD(0,fibonacci1)
    LOAD(1,fibonacci2)
    ADD_R(0,1,0)
    STORE(0,fibonacci2)
    LOAD(0,temp)
    STORE(0,fibonacci1)
    LOAD(0,count)
    LOAD_IMMEDIATE(1,1)
    ADD_R(0,1,0)
    STORE(0,count)
    LOAD(0,count)
    LOAD_IMMEDIATE(1,10)
    COMPARE(0,1,0)
    MOV_R(0,1)
    MUL_R(0,1,0)
    JUMP_POSITIVE(0,repeat_begin_1)
repeat_end_1:
    STOP
```

```
count = 10
fibonacci1 = 55
fibonacci2 = 89
temp = 55
```

```
Μεταγλώττιση Επιτυχής
```

```
Οι πρώτοι 10 αριθμοί Fibonacci
είναι:
1,1,2,3,5,8,13,21,34,55,
Ok
```

Figure 1. The WIPE Compiler environment

3.3 Pseudo-assembly

The WIPE Compiler takes the source code and produces a pseudo-assembly code that runs in a virtual machine with two registers and a stack. More advanced programmers can activate the assembly code window and observe the relationship between the source and assembly code by executing their program in a step-by-step fashion. In Figure 1, the top right window contains the assembly code that corresponds to the source code of the top left window. An additional tab in the variable watch window (bottom left) displays the intermediate values of the virtual machine registers during program execution. Notice the various controls beneath both the source and assembly windows.

Of course, we do not recommend the activation of the assembly window for novice programmers. This is a feature that teachers could use in certain didactic situations or motivated students could explore (textbooks in Greek secondary schools cover some basics of assembly programming).

The use of an intermediate pseudo-assembly code was a design decision. It was adopted so that WIPE becomes rather easily extensible. By adding additional source language to pseudo-assembly compilers, WIPE can support for additional source languages (e.g., C/C++, Java).

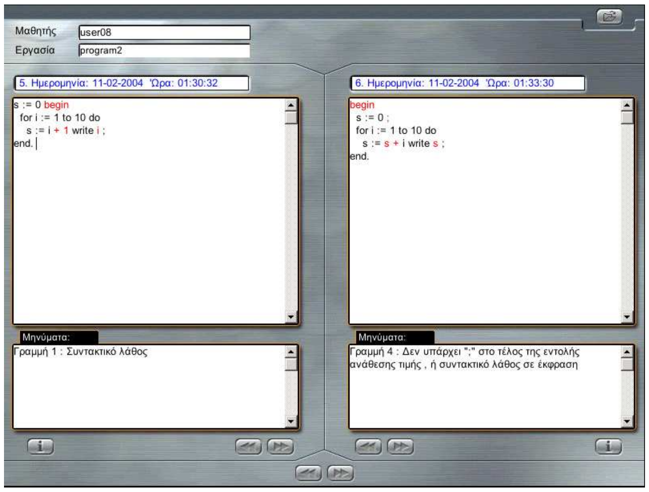
3.4 Learning Aids

WIPE implements a series of features that simplify the learning process for novice programmers.

- The implemented source programming language in WIPE does not require variable and constant declarations. Programmers do not have to worry about declaration statements.
- Error messages are not cryptic, a common problem in commercial compilers. We tried to make them precise and also included hints for possible solutions.
- WIPE provides run-time error detection. The programmer is informed about the presence of run-time errors, such as, division by zero, use of variables that have not been initialized, and, operand type checking in the case of arithmetic operations like *mod* or *div*.

3.5 Recordability

A very important and novel feature of WIPE is the recording of all student actions, i.e., recordability. A series of student actions, such as, compilation, program execution, and step-by-step execution, are being recorded in a special database. All that data are readily available to the teacher either as raw data from the database or via the use of specially designed tools. These tools help the teacher visualize the data and infer useful information. For example, see Figure 2 where the teacher can compare successive compilations of a student program. Differences of successive versions of the code of the program are pinpointed as well as the corresponding compiler errors.



Μαθητής user08
Εργασία program2

5. Ημερομηνία: 11-02-2004 'Ώρα: 01:30:32

```
s := 0 begin
 for i := 1 to 10 do
   s := i + 1 write i ;
end.
```

Γραμμή 1 : Συντακτικό λάθος

6. Ημερομηνία: 11-02-2004 'Ώρα: 01:33:30

```
begin
  s := 0 ;
  for i := 1 to 10 do
    s := s + i write s ;
end.
```

Γραμμή 4 : Δεν υπάρχει ";" στο τέλος της εντολής ανάθεσης τιμής , ή συντακτικό λάθος σε έκφραση

Figure 2. Code version visualization tool for teachers

3.6 Didactic Situations

Perhaps the most valuable element in the use of educational software is not the software itself but its usage. This is what the modern theory of didactics of Mathematics identifies by the term *didactic situations*. Since this rationale appears to be valid not only in Mathematics but also in the teaching of Programming (which according to some researchers is a branch of Mathematics [14]), we use WIPE in the context of pre-designed didactic situations. These didactic situations include select problems that are tackled under certain conditions in order to address and reveal the usual misconceptions of novice programmers (e.g., anthropomorphism, the role and usage of variables, etc. [20]).

3.7 Implementation Details

The WIPE components were implemented using: (a) C and the compiler generation tools Flex and Bison, for the Pascal-like language Compiler, (b) PHP, for the web pages and the communication between client and server, and, (c) Macromedia Director (Shockwave output), for the front end on the client side. Also, we used the open source DBMS MySQL. A demonstration of WIPE is available at http://eos.uom.gr/~efop.

4. EVALUATION

In the end of 2003 and the beginning of 2004, WIPE was tested in three secondary education schools (1st Gymnasium of Asvestochori Thessaloniki, 1st Technological and Vocational School of Argos Orestiko Kastoria, and 1st Lyceum of Nestorio Kastoria).

Forty-five students were taught six two-hour long laboratories using WIPE during the regular teaching hours of computer science related courses.

The collected data came from:

- Classroom observation during the laboratories.
- Analysis of the student programs and their actions that were recorded in the WIPE database.
- Programming environment evaluation questionnaires that were completed by the students after the completion of the six laboratories.
- Interviews with the teachers that were responsible for the software evaluation process.

In the following, we briefly report on the most important findings of the evaluation.

An analysis of the source code written by students reveals two categories of errors: syntax and logic errors. Table 1 summarizes the most common syntax errors that were detected:

Table 1. Syntax errors

typo errors
missing "end" statement
missing keywords "do" in structures while .. do and for .. do "then" in structure if .. then .. else .. endif
missing ";" at the end of each statement
missing brackets and quotes
missing comma in a write statement (between two expressions)

These errors could be attributed to the particular syntax requirements of the WIPE source language, which differ from the syntactic conventions of a natural language that allows great flexibility (for example the end of the condition in a natural language arises from the meaning of statement and not from some specific keyword such as "endif").

Table 2 summarizes the most common logic errors that were detected.

Table 2. Logic errors

failure to increment a loop counter variable (infinite loops)
erroneous incrementing of a loop counter variable (i.e., outside the loop)

erroneous prompting (the user is asked to enter a value for a variable after the value has been read). For example:
read number;
write "Enter a number: ";

accessing of non existing array elements

Our conclusions from the testing and evaluation process match the didactic goals we set during the design phase of our educational software. We briefly list them below:

- The programming environment and the accompanying educational material (didactic situations) greatly assisted the students to understand the fundamentals of programming. This conclusion results from the answers of students in the evaluation questionnaire but also from the analysis of the information recorded in the database. The majority of students (about 80%) consider the accompanying educational material very helpful for acquiring new knowledge and better comprehending programming.
- The students did not hesitate to use the programming environment and their adaptation was easy. The programming environment proved to be user friendly and functional and in general stirred the interest of students. Most of them wish to use WIPE while studying programming on their own (about 81%). They would also welcome the option to have such kind of software be distributed together with their programming language textbooks (about 78%).
- The WIPE compiler messages were simple and comprehensible for the average student. Most students consider the messages comprehensible and believe that they facilitate the detection of errors (average 4 in a 1-5 scale), while 86% of them classified the error messages among the characteristics of the programming environment that satisfied them the most.
- Of course, no programming environment can eliminate the errors novice programmers make. WIPE, however, assisted students in easily correcting their errors by having the compiler provide comprehensible error messages, by using coloring for keywords, by implementing a step-by-step source code execution feature, and by displaying the intermediate values of variables. The recorded data reveal that students use very often the step-by-step execution feature (more than 50% of program runs are step-by-step executions). That particular feature is evaluated positively as one of the features of the environment that satisfied the students (about 83% are in favor), while 94% of the students appreciate the feature of displaying the intermediate values of program and system (temporary) variables and registers.
- The teachers that used the environment for teaching programming were cautious in the beginning. But in the course of the evaluation they started making positive comments and expressed their satisfaction for being able to use this educational environment. The teacher tool was a surprise for them and they were impressed by the wealth of information that it provides. Our final conclusion was that WIPE constituted a useful tool for the teachers for teaching the fundamentals of programming to novice students.
- The accompanying educational material (a series of carefully designed exercises to be used as didactic situations) supported the teachers during the teaching process and provided them with a productive way of using the programming environment. As far as the students are concerned, those exercises helped them become familiar with the environment, learn how to use it, and make proper and productive use of its various features. These didactic situations offer students directions for better exploitation of the provided features. The average ranking for the educational material was 3.5 (scale 1 to 5).

5. CONCLUSION

Although educational environments for teaching programming appear to have clear advantages over the commercial solutions, almost all of the currently available such environments have a hard time trying to establish themselves. There are many obstacles they should overcome, namely, the commercial competitors that strongly promote their own solutions, the perception of novice programmers that hesitate to use and doubt the usefulness of a non-brand name environment that nobody uses in the real world, and finally the teachers that resist change and prefer to use tools they are already familiar with.

WIPE is an educational environment for teaching programming whose design is based on the accumulated experience and practice gained from numerous related research efforts in the broader area of the teaching of programming. Its application in the school setting has revealed some operational and didactical shortcomings, but we claim that the overall design and implementation process that was followed is satisfactory. We expect that WIPE will be used as a means for directly or indirectly collecting data that will form the basis for a more systematic study of the issues related to the teaching of programming. At any rate, regardless of the didactic means used, we claim that an improvement in the teaching practices of programming and computer science in general has to use a unifying framework that will sort out the various results and gathered data, establish successful methodologies, and verify research findings.

6. ACKNOWLEDGMENTS

Our thanks to the teachers and students that evaluated WIPE.

7. REFERENCES

[1] AACE, Association for the Advancement of Computing in Education, http://www.aace.org/default.htm

[2] ACM SIGCSE, ACM Special Interest Group on Computer Science Education, http://www.sigcse.org

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

[4] Brooks, R. Towards a theory of the cognitive processes in computer programming. In *International Journal of Man-Machine Studies* 9, 1977, 737-751.

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

[6] Brusilovsky, P., Calabrese, E., Hvorecky, E., Kouchnirenko, A., & Miller, P. Mini-languages: A Way to Learn Programming Principles. In *Education and Information Technologies*, 2(1), 1999, 65-83.

[7] Computer Science Education Journal, Taylor & Francis, http://www.tandf.co.uk/journals/titles/ 08993408.asp

[8] Conway D. *Criteria and Consideration in the Selection of a First Programming Language.* Technical Report 93/192, Department of Computer Science, Monash University, December 1993.

[9] Dagdilelis V. *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, 1986.

[10] Dagdilelis, V., Evangelidis, G., Satratzemi, M., Efopoulos, V., Zagouras, C. DELYS: A novel microworld-based educational software for teaching Computer Science subjects. *Computers & Education*, Elsevier Science, 40(4), May 2003, 307-325.

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

[12] Eisenstadt, M., Lewis, M. W. Errors in an Interactive Programming Environment: Causes and Cures. In *Novice Programming Environments: Explorations in Human-Computer Interaction and Artificial Intelligence*, Marc Eisenstadt, Mark T. Keane, and Tim Rajan, (eds), Lawrence Erlbaum Associates, Hillsdale USA, 1992.

[13] Evangelidis, G, Dagdilelis, V.,Satratzemi, M., Efopoulos, V. X-Compiler: Yet Another Integrated Novice Programming Environment. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies* (ICALT 2001), 166-169, Madison, WI, USA.

[14] Hoare C.A.R. Is there a mathematical basis for computer programming? *NAG Newsletter*, 2, 6-15, 1981.

[15] International Journal of Human-Computer Studies, Elsevier, http://www.elsevier.com/wps/find/journaldescription.cws_home/622846/description#description

[16] ISTE, International Society for Technology in Education, http://www.iste.org

[17] Kölling, M. Teaching Object Orientation with the Blue Environment. *Journal of Object Oriented Programming*, 12(2), May 1999, 14-23.

[18] McIver, L., Conway, D. GRAIL: A Zeroth programming language, In *Proceedings of the International Conference on Computing in Education* (ICCE99), 43-50.

[19] Murnane J. The Psychology of Computer Languages For Introductory Programming Courses. *New Ideas in Psychology*, 11(2), 1993, 213-228.

[20] Pea, R. D. Language-independent conceptual bugs in novice programming. *Journal of Educational Computing Research*, 2(1), 1986, 25-36.

[21] PPIG, Psychology of Programming Interest Group, http://www.ppig.org

[22] Weinberg, G.M. *The Psychology of Computer Programming*. New York, Van Nostrand Reinhold, 1971.