# A Visual-Textual Programming Environment for Comparative Studies of Programming Tasks

**Maria Beatriz Serrano Valdivia**
1. Universidad de Carabobo - Facultad de Ingeniería
Departamento de Computación – Valencia - Venezuela
+58 (241) 867-0561
mserrano@uc.edu.ve
2. University of Massachusetts Lowell - Computer Science Department
Lowell, MA 01854

**ABSTRACT**
In this article, we describe a programming environment with equivalent functionality for textual and visual representation of programs. This environment is designed with the aim to rule out confounding variables for comparative studies of programming tasks. To aid in empirical studies, the environment has the additional capability of recording times for the KLM operators: typing a character and click mouse button. It also records the time for mouse drag and drop actions. This preliminary study suggests that the programming environment is suitable for empirical comparative studies of programming tasks and to be modeled by formal task analysis methods such as NGOMS.

**Keywords**
Programming environments, Visual and Textual program representations, GOMS, Task Analysis.

## INTRODUCTION

Several comparative studies of programming tasks have been reported. Most of these studies, including the often-cited comparisons of visual and textual programming tasks by Green and Petre [3, 6], have been "apples and oranges" comparisons because of the presence of confounding variables, as discussed by Moher, Mak et al. and Williams, Villegas et al. [4, 8].

This area of research involves three main issues: equivalent representations of visual and textual programs, definition of programming tasks that can be modeled, and selection of a task analysis method. We center this study on designing an environment that provides and equivalent representation of programs with equal functionality.

### Graphical vs. Textual Program Representation

Recent research comparing graphical vs. textual program representation has been questioned because the graphical programs have not been optimized as the textual programs. [4,8]. This could be because the graphical and textual program representations used in the studies were not equivalent. The environment presented intends to address this issue, by providing a common interface and equivalent representations (visual and textual) of a program.

Based on the suggestions that the appropriateness of a graphical representation may be task-specific and that ease of reading a graphical program depends on layout.[4,8] we decided to use flowchart figures to display the graphical program representation because they directly symbolize the program constructs that a general purpose programming language supports.

### Programming tasks

Not all programming tasks can be modeled nor all programmers solve problems the same way. The history of the psychology of programming dates back to the 1960s, to the work of Rouanet and Gateau, who analyzed business programmers' behavior in terms of a top-down, data-oriented programming methodology [7]. Since then, many attempts to describe the different activities, the psychology and cognitive processes behind the activities involved in programming are found in the literature. In general, researchers coincide in the fact that programming is a set of tasks a programmer performs to write a program, although they might not name or characterize them the same way.

These tasks may be characterized in three basic stages: understanding the problem, coding and documenting the program, and maintenance of the program. Pennington and Grabowski report on the required knowledge to perform each task. They conclude that programming is a complex activity that involves cognitive processes and several kinds of specialized knowledge [5]. Although, Gilmore and T. R. G. Green [1, 2] report on related programming issues, they mention that the basic tasks involved in programming are designing, writing, reading, and debugging programs.

Given the diversity of activities and psychological processes involved, we expect that this environment will help determine which programming tasks can be modeled by studying the programmer design specific simple programs.

## THE ENVIRONMENT

The environment designed has the capability of creating, editing, and executing visual and textual representations of a program. As a user is working on the visual or textual window, the actions are updated on the other representation. For example if the user is inserting a while-loop in the textual representation, the graphical figure for the loop is inserted simultaneously on the visual representation. Because of this characteristic the user can change windows [representations] easily.

### Programming language implemented

The interface is designed as a front-end tool to interact with the programmer (user). At the moment, the programming language implemented is Pascal, but it has the capability of adding and selecting other general purpose programming languages, for example: C. The environment provides a configuration window, shown in Figure 1, which presents the option of adding a new language by specifying the reserved words that define the language, selecting the data types the language supports and adding the libraries to be included.

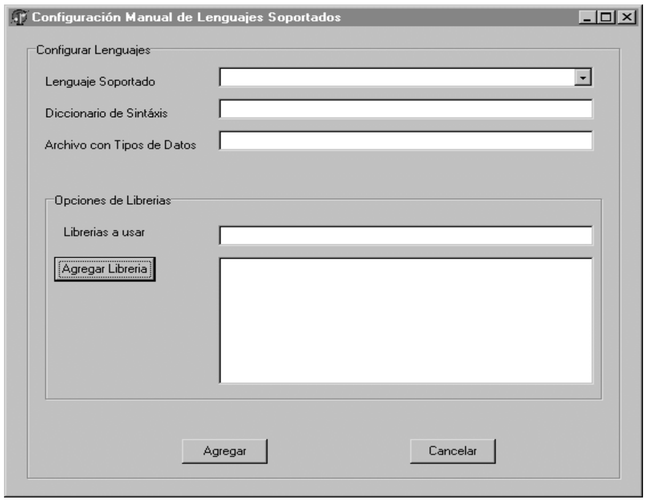
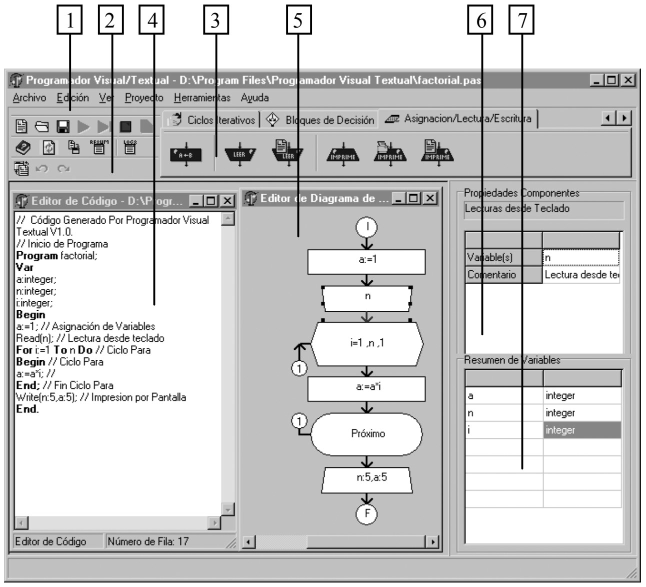


**Figure 1: Snapshot of the language configuration window**

### Description of the interface

Figure 2 shows the seven areas that conform the interface: (1) the general File, Edit, View, Tools, Help Menu Bar; (2) the Action Button Area to compile and execute programs, refresh or change active program representation, and present time and activity log windows; (3) the Program Construct area that presents the flowchart figures to be selected organized in tabs for Program Start/End, Input, Output, Selection, Loop, and Assignment constructs; (4) Textual Program Representation; (5) Visual Program Representation; (6) Statement specifications and (7) Variable declaration.



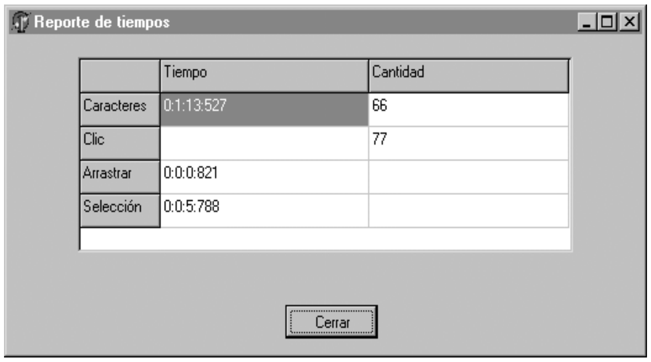
```
// Código Generado Por Programador Visual
Textual V1.0.
// Inicio de Programa
Program factorial;
Var
a:integer;
n:integer;
i:integer;
Begin
a:=1; // Asignación de Variables
Read(n); // Lectura desde teclado
For i:=1 To n Do // Ciclo Para
Begin // Ciclo Para
a:=a*i; //
End; // Fin Ciclo Para
Write(n:5,a:5); // Impresion por Pantalla
End.
```

| | |
|---|---|
| Variable(s) | n |
| Comentario | Lectura desde te |

| | |
|---|---|
| a | integer |
| n | integer |
| i | integer |

**Figure 2: Snapshot of the programming environment: Program solved: Determine n!, for 0 < n < 8**

In the back-end, the textual representation of the program is compiled and executed by a standard compiler; in this case the Pascal compiler used is FreePascal. When saving the program only the textual program is saved, the visual representation is created at the moment of loading the program.

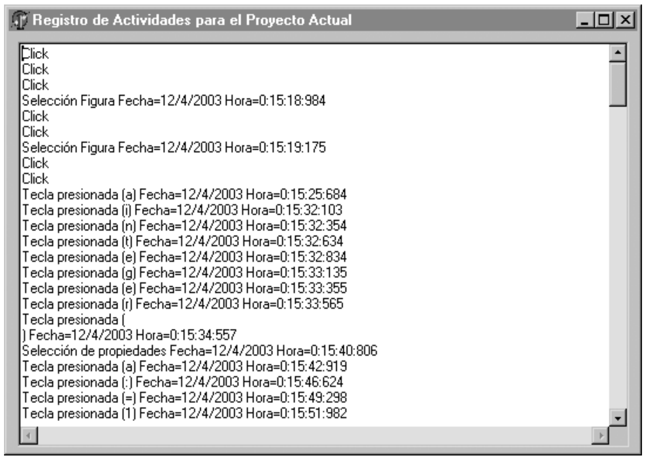


| | Tiempo | Cantidad |
|---|---|---|
| Caracteres | 0:1:13:527 | 66 |
| Clic | | 77 |
| Arrastrar | 0:0:0:821 | |
| Selección | 0:0:5:788 | |

**Figure 3: Summary of Activities Performed**

### Time and Activity Log Windows

Figures 3 and 4 show the time and activity log windows that record the users actions. This information is useful for the future evaluation of empirical studies to be performed. The activity logs will support the NGOMS model in how a programmer performs specific tasks and the time logs will support the empirical studies in determining execution and learning times.



```
Click
Click
Click
Selección Figura Fecha=12/4/2003 Hora=0:15:18:984
Click
Click
Selección Figura Fecha=12/4/2003 Hora=0:15:19:175
Click
Click
Tecla presionada (a) Fecha=12/4/2003 Hora=0:15:25:684
Tecla presionada (i) Fecha=12/4/2003 Hora=0:15:32:103
Tecla presionada (n) Fecha=12/4/2003 Hora=0:15:32:354
Tecla presionada (t) Fecha=12/4/2003 Hora=0:15:32:634
Tecla presionada (e) Fecha=12/4/2003 Hora=0:15:32:834
Tecla presionada (g) Fecha=12/4/2003 Hora=0:15:33:135
Tecla presionada (e) Fecha=12/4/2003 Hora=0:15:33:355
Tecla presionada (r) Fecha=12/4/2003 Hora=0:15:33:565
Tecla presionada (
) Fecha=12/4/2003 Hora=0:15:34:557
Selección de propiedades Fecha=12/4/2003 Hora=0:15:40:806
Tecla presionada (a) Fecha=12/4/2003 Hora=0:15:42:919
Tecla presionada (:) Fecha=12/4/2003 Hora=0:15:46:624
Tecla presionada (=) Fecha=12/4/2003 Hora=0:15:49:298
Tecla presionada (1) Fecha=12/4/2003 Hora=0:15:51:982
```

**Figure 4: Listing of activities and times recorded**

## FUTURE WORK

The next phase is to evaluate the environment by repeating Green et al.'s study and other programming tasks. We expect to determine the usefulness of the environment for comparative studies of visual and textual programming tasks by comparing the NGOMS models versus empirical studies. With these comparative studies results we expect to determine which programming tasks can be modeled and understand how novices [non computer science students] understand a problem and then write a program to solve it. This issue is part of our continuing research.

## ACKNOWLEDGMENTS

We thank the students Jesús Hernández and Leandro Pacheco, from Universidad de Carabobo, that helped develop this research tool.

## REFERENCES

1. Gilmore, D. J. (1990). *Methodological Issues in the Study of Programming*. San Diego, California, Academic Press, pp. 83-98
2. Green, T. R. G. (1990). *The Nature of Programming*. Psychology of Programming. San Diego, California, Academic Press, pp. 21-44.
3. Green, T. R. G., M. Petre, et al. (1991). *Comprehensibility of Visual and Textual Programs: A Test of Superlativism Against the 'Match-Mismatch' Conjecture*. Empirical Studies of Programmers Fourth Workshop, Ablex Publishing Corporation: 121-146.
4. Moher, T. G., D. C. Mak et al. (1993). *Comparing the Comprehensibility of Textual and Graphical Programs: The Case of Petri Nets*. Empirical Studies of Programmers Fifth Workshop, Ablex Publishing Corporation: 137-161
5. Pennington, N. and B. Grabowski (1990). *The Tasks of Programming*. Psychology of Programming. San Diego, California, Academmic Press, pp. 45-62.
6. Petre, M. (1995). *Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming*. Communications of the ACM. 38: 33-44.
7. Rouanet, J. and Y. Gateau (1967). *Le travail du Programmeur de Gestion: Essai de Description*. Paris, AFPA-CERP
8. Williams, M. G., H. Villegas, et al. (1996). "Appropriateness of Graphical Program Representations for Training Applications." Conference Companion of the ACM SIGCHI 1996 Conference of Human Factors in Computing Systems: 91-92.