Olympiads in Informatics, 2015, Vol. 9, 27–37
DOI: http://dx.doi.org/10.15388/ioi.2015.03

Methodology for Characterization of Cognitive Activities when Solving Programming Problems of an Algorithmic Nature

Gilberto CUBA-RICARDO, María T. SERRANO-RODRÍGUEZ,
P. Alberto LEYVA-FIGUEREDO, Laura L. MENDOZA-TAULER
José de la Luz y Caballero University of Pedagogical Science of Holguín. Cuba
e-mail: {gilberto.cuba, mariat, albertoleyva, laura}@ucp.ho.rimed.cu

Abstract. This paper shows a methodology for characterization the students' cognitive activity when solving programming problems of algorithmic nature. It also reveals the methodology stages and the dimensions assumed to assess the process using several methods, techniques and tools.

The paper describes some characteristics and regularities from the behavior of three students when they solve a programming problem.

Keywords: programming contests, problem solving, metacognition.

1. Introduction

Programming contests are constituted by problems that contestants must solve in a short period of time in relation to its level of complexity (Verhoeff, 1997). Hence, some of the results obtained are less than what it is expected from a student.

The informatics coach must diagnose the students' development stage when mastering these algorithms. They must also diagnose their skills when solving these kinds of problems. These elements constitute a challenge from a pedagogical point of view. As it is important for the coach to influence and improve students' work, identifying their deficiencies is critical.

The source code constitutes the solution given by the students to a specific problem. The worksheets for such contestant are the space where they write some models or techniques related to the problem. This source code and their worksheets don't allow for the complete analysis of their skills and shortcomings. This is because the elements are results of the final state of a process lasting between 60 and 90 minutes. Consequently, a current issue is the study of the cognitive activity developed by the student when solving programming problems (Deek *et al.*, 1999; White and Sivitanides, 2002).

Specifically, Hosseini *et al.* (2014) show a new approach for assessing the exercise solutions. Here, they check the intermediate steps during implementation of an algorithm solution. This alternative allow Hosseini *et al.* (2014) to conclude that the study of these intermediate steps helps the coach to know the most common paths used by students when solving programming problems, and thus, provide better feedback to students. However, these authors only consider at each intermediate step, the source code developed by the student.

Surakka and Malmi (2004) list the cognitive skills that successful computer programmers must have when solving programming problems. These cognitive skills are the results of many experienced programmer opinions through the application of two rounds of the Delphi method. However, programmer opinions were not contrasted with information gathered from other practical methods.

In such case, the study of cognitive activity when students solve programming problems should not only consider the analysis of the aforementioned process results, but also the assessment of the thinking states of the students while they solve the problem.

In this sense, considering the possibilities offered by the computer to register the interactions given by the user in short periods of time, it is decided then to search for an alternative to register the students' behavior while solving programming problems. In addition, a methodology was developed to guide the process considering some process and outcomes indicators.

The methodology was applied to three senior high school Informatics contestants.

2. Screenshot Software

When seeking informatics applications that met the register requirements, some were found which were oriented to monitoring and registering the traces left by the user during their interaction with the computer. Their usage is supported by a conception involving and compromising the computer's security for malicious purpose. It is very common to recognize them as malware, or relate them to monitoring processes in enterprises that monitor informatics security. However, the application explained below, has different purposes in relation to the scientific research, mainly, to reveal behavior when solving programming problems of algorithmic nature.

Without going further on the security matter, some of the names that identify them are *Spyware, Spybot, Keylogger, Computer Monitor, PC Audit*, etc. Almost all of those found are registered under owners' license and with a great amount of setting options which slow the output of the computer. Plus, it makes usage for more specific purposes like only recording screenshots in short periods of time difficult.

Thus, a very simple informatics application was developed, which allowed the accomplishment of the stated objectives without reinventing those that already have their established goals related to the malware or security. Nevertheless, the core of the research wasn't the development of a new sophisticated application that permitted the registration of the entire user's interaction with the computer. Up to this moment, we

have only implemented the registration of snapshots of the computer desktop state over short periods of time.

The name given to this application was *Screen Shooter System Tray*. It has an executable that can be instantiated from any location in the computer. Installations requirements do not depend on frameworks or third party libraries to allow execution. It only runs on Microsoft Windows XP or later versions.

Its visual recognition is presented as an icon in the computer's operating system's tray, next to the clock. It has a context menu with the basic options to enable and disable the recording. After the execution of the application, the image recording is saved by default in the same folder of the executable file, with a one-second difference between screenshots.

Since the number of images saved in one hour is greater than 3600, it is necessary to reduce image sizes. For that, images are saved as JPEG format in grayscale with 8 bits depth of colors.

3. Methodology for Characterizing the Cognitive Process of Solving Programming Problems

The methodology used was not only aimed at a final assessment of such a process. There are several manual methods to do so, allowing checking the solution of an exercise with the data sets produced with the objective of getting a grade. There are also Internet Websites, that devote their content to publishing exercises and evaluating the solutions submitted by the users. (Kolstad and Piele, 2007; Revilla *et al.*, 2008; Verhoeff, 2008; Mares and Blackham, 2012; Maggiolo *et al.*, 2014).

The objective of the present methodology was to determine the characteristics shown by students during the programming problem solving process, at the time they evaluate the quality of the final outcome, expressed in the source code of the problem's solution algorithm. For a better understanding and materialization of it, the methodology was organized into five stages:

1. Preparation for the process.
2. Recording the process of exercise application.
3. Analysis, processing and assessing of the partial reports (observation, desktop pictures, source code of the algorithm solution).
4. Interview session.
5. Final assessment.

In this regard, deep studies on the characterizations of the student's cognitive activity while solving problems have no recent antecedents in the area of mathematics in the works of Shoenfeld (1985) and Cruz (2002). On this basis, common features have been analyzed and certain procedures, which characterize the programming problem solving process, have been imported from Mathematics.

In Informatics, specifically in the Programming area, the work of Deek *et al.* (1999) has been found. These authors developed their research in the introductory courses of

problem solving and programming taught to university students from the New Jersey Institute of Technology. They used a method to assess processes developed by the students, which allowed them to readjust the content and related teaching methods, supported by a six-stage model.

Their arguments allowed us to clarify the variables to consider in the characterization process, and we used some of the indicators presented in their work. However, their instruments are not used, since they do not take advantage of the interview potential with each student.

Specifically, for the implementation of the methodology, we recognized that training and developing metacognitive processes in the students, has an important role in the solving programming problems of an algorithmic nature. Hence, one of the dimensions taken into consideration was *metacognition*. The other one was the use of *complementary tools* that help the contestant solve the problem. The third one was *solving the stated problem*, which is a source code in a Programming Language (PL) reflecting an individualized representation of the solution algorithm.

The first two dimensions have a process nature, while the third one is a result. This demonstrates the objective of the methodology, essentially aimed at the characterization of solving programming problems of an algorithmic nature.

3.1. *Stage (I) Preparation for the Process*

A careful selection of the exercise was essential to carry out the characterization process. For that, the previous knowledge of the students was taken into consideration, along with what it is needed from the problem itself to find a solution algorithm. It was also guaranteed that there were several solutions of the problems and the data sets which allowed us to assess the students' answers.

The exercise was prepared to be read by the student from a document reader in each computer. This permitted us to know through screenshots when the contestant consulted the document for the exercise. Furthermore, the computers' clocks were synchronized when applying the same programming problem at the same time.

When collecting the data which enables characterizing the process, three main methods were used: registration of participant observations, the screenshot recording and the interviews. The first two were developed while solving the problems and the last one, once the process was completed.

The purpose of the observation in the development of the activity, was aimed at describing in as much detail as possible, the students' behavior when solving problems. Hence, the registration of the observation focused on writing what the students did in every time interval. For that reason, it was designed as a three-column table containing the description of the action, including the start and end time.

While solving the problem the student took some notes and reflected. These reflections are show when he is not reading the exercise or not writing on the worksheet. To be sure of the second action, the students were asked some questions.

3.2. *Stage (II) Recording the Process of Exercise Application*

This activity began using the proposed application that captures the computer screenshots used by the students (*Screen Shooter System Tray*), at the time it gave them the start command. From then, every observation moment was registered on worksheets. The students' doubts were also assisted regarding the interpretation of the problem, which were also registered like the other actions. Doubts were essential elements to determine the students' comprehension levels in relation to the exercise. This obviously helped orient the questions of the interview.

With a little difficulty some of the students' behavior was also registered such as anxiety, despair, uneasiness, satisfaction, success, joy, fear, defeatism, etc, which allowed a comprehensive assessment of the process. Some of these behaviors were identified through questions, because it was difficult to diagnose from the observation.

Likewise, the students were told when they had 5–10 minutes left to finish the exercise like in any other contest of this kind. At this point, the students' decisions changed, which were also registered as much as possible in order to fulfill the assessment strategy developed by them when solving problems under pressure.

When the time dedicated to do the task finished, the application recording the screenshots was stopped. The images were collected along with the programs source code to be processed. Worksheets were also gathered and added to each student's observation sheet.

The students were not allowed to tell each other their experiences until the interview process was over. This decision was mainly based on the fact that they could readjust not only their solving algorithm, but their behavior and opinions. It was considered that if this exchange of ideas could happen, the results of the students' answers in the interview would have altered.

3.3. *Stage (III) Analysis, Processing and Assessing of the Partial Reports*

It was decided that the time for the problem solving process, and the interview sessions wouldn't exceed 24 hours. This was based on the following: the larger the time between the two activities, the less the possibilities to really know the characteristics and cognitive behaviors developed by the students when solving the problems. If the time exceeded 24 hours, the students could forget why they determined certain decisions that were revealed during the observation process. Therefore, once the recording process was finished, every screenshot was further analyzed, and the interview began right after that, with the objective of clarifying those elements that were not very clear in the interpretation process, and to clarify some hypothesis that surfaced from the analysis process.

When assessing the previously declared dimensions, some indicators were taken into consideration that enriched the process in its abstraction. In the metacognitive

dimension, the indicators were focused on: the metacomprehension of the problem and the given solution, its planning, the conscious use of strategies and techniques to solve the problems, the self assessment of the steps followed and the whole process as such.

In relation to this foundation, the studies of Weinstein and Underwood (1985) were consulted. They proposed questionnaires representing inventories dedicated to assess the learning strategies, which, among others, metacognition can be found. The review of such questionnaires permitted formulating a guide to questions to be asked in the interview stage.

Most of these indicators were best suited to the interview stage. However, from the analysis of the desktop screenshots, along with the students' notes in the worksheet, plus the notes from the observation, the guide of questions and topics to ask during the interview were planned. As an example, some of the topics and questions asked of the students in the interviews were:

- Could you read and fully understand the exam exercise? How and/or when do you know you had correctly understood the exercise?
- Was the exercise difficult or easy to answer? Why? What elements did you take into consideration to determine the difficulty of the exercise? What was more difficult: understanding the exercise or searching for an algorithm is solution that fulfilled the requirements of the exercise, and/or to code in a PL the algorithm solution found?
- Did you feel ready to solve this kind of exercise? Was there any kind of content you had not mastered and felt you needed to solve the exercise? Which one? What do you do in such situation, when there is an exercise you cannot completely solve during a contest?
- Did you use any particular strategy to solve the exercise? Can you tell us about it? Have you yourself defined any steps to follow, or that you had already planned beforehand to solve the exercise? What techniques, tools, steps or ways did you use to solve the exercise successfully?
- When you found an algorithm is solution for solving the problem, how did you know it was correct to start to code it and that it will be successful? Or did you just start coding it and check at the end with the data sets if it is right or wrong?
- Was the exercise similar to others previously answered by you during the training period or in any other contest?
- What were your notes in the worksheets used for? Can you explain some of them?

Most of these questions have their own purpose. The first one, deciphers the characteristics of the students' thought, along the problem solving process; the second one, educational, evidenced when the student is introspective and consciously gains the knowledge of how certain processes are developed in his thought. Therefore, these interviews were very important because they make possible the development and training of metacognitive skills in students.

When analyzing the dimension related to the additional tools that enabled solving the

problem, it was considered that: the computer's operating system, the Integrated Development Environment (IDE), debugger and PL, are part of this group. When recognizing the skill level of the students' development while using these tools, the understanding of their basic concepts is also taken into consideration.

Some problems detected after the observation and the analysis of the screenshots, are presented below:

- The keyboard configurations do not allow a suitable output of the symbols needed to code the algorithmic solution in the selected language.
- Inadequate mastery and familiarization of IDE, which makes the necessary operations impossible to code with the required speed.
- IDE configurations do not facilitate a larger visibility of the source code and access to the most used objects.
- The debugger possibilities were wasted.
- Insufficient mastery of the PL used to code the algorithm solution, mainly evidenced when the student doesn't know about the range of the data types, when he doesn't recognize the syntax errors indicated by the compiler, and when he doesn't interpret the functionality of part of the program code.

Finally, when analyzing the final solution of the problem as a source code of such an algorithm, the following were taken into account: the total grading obtained during the program assessment, the source code style, the relations between the source code and the algorithm solution, and the data structure in their relation to code portions supporting the input, output and processing of the data.

The clarity of the source code, as a manifested characteristic of the solution program submitted, is related to the planning done by the student during the initial process of the solution conception. Consequently, the analysis of its links and the search of explanations by the students is needed. Its effectiveness was checked through the assessment of the program in execution with every data set.

As the final solution of the exercise was not very clear, it was necessary to investigate some matters related to the source code and the solution algorithm found. Upon this foundation, the following questions were planned:

- Explain briefly the solution algorithm found for the exercise. First of all, express it in a mathematical way according to the data and the unknown data. Then, explain yourself, demonstrating in praxis the use of the data structures, as well as the control structures used in the programming language.
- How did you determine the data structures you were going to use? Which do you determine first, the structures or the algorithm?
- When you started to code the program, did you do it thinking about the solution algorithm of the exercise? Did it change during the time used to solve it? Can you describe the main differences or changes produced between the algorithm as first conceived and the one delivered? How do you know if the coded program solves the exercise?

Once the interview guide is organized, it is added to the source code, the students' worksheet and the observations made in order to move on to the next stage.

3.4. *Stage (IV) Interview Session*

The interviews were individual and clarified the hypothesis and doubts observed from the students' behavior. The need to have devices that allow recording videos become evident. It was also verified, that during the activity other questions surfaced that should have been planned and arranged as part of the interview guide. Other matters that prevailed were the students' exhaustive explanations about the algorithm they followed, and the way they took it to the "new" worksheets.

At the end of the interview, the student was shown the algorithm planned as the solution to the exercise in case it did not match there, and they were asked for opinions about it. Likewise, the relation between the presented algorithm and the source code that solves it was explained. During this explanation, he was also shown some modeling techniques that enable a better representation of the information of the problem and the relations established between them.

The exemplification of this element not only shows to the student the way to solve the stated problem, as the interview not only gives the teacher those elements he must teach in order to improve the solution of the programming problem; but also, it allows the student to think of his metacognitive activity, of his techniques and strategies applied to solve the problem, and of his skills. In general, this kind of procedure influences the student self regulation from the very moment he knows how his knowledge is developed.

3.5. *Stage (V) Final Assessment*

Reaching this point, an exchange cycle with the contestant is closed, where most of the elements explained by him are corroborated with his worksheet, along with the registration of the observation.

In order to continue clarifying the problem solving process developed by the students, and after a first screenshot review, a further and exhaustive review was undertaken, where some elements were revealed constituting problems, wasting of time or difficulties that obstruct the speed of the problem solving process.

In general, after the analysis of several of the represented data, we concluded that the students demonstrated a tendency to think and solve the problem as they coded the solution algorithm. This aspect confirms the strong union between the construction of the problem solution algorithm and its code. (Deek *et al.*, 1999).

This process is evidenced when the student starts to code the solution of the problem; reinterprets the written code and compares it to his algorithm solution represented in his mind. Then, he corrects the possible mistakes little by little, and optimizes the source code in correspondence to the planned objectives.

This phenomenon is shown as an unfinished idea, and in the way the student implements the source code, he evaluates and controls mentally its parts. This interpretation is carried out through internal simulations, and such portions are closely related forming little algorithms that in its general structure, determine the algorithm solution of the exercise presented.

Taking into account this behavior, it was observed that students have difficulties when interpreting the problem, as well as when planning the search for the solution. Similarly, it was evidenced that leading the heuristic search in the construction of the solution algorithm, was made unconsciously and not in a self controlled way, which would facilitate the correct selection of the strategies to solve problem of this nature.

Throughout the process, it was also shown, that the students check the partial status of the functioning of the source code implemented, for example: data reading and writing, small algorithm of filling data, preprocessing and simple algorithms expressed as subroutines.

It was also concluded that the students demonstrate, in some cases, the classic use of trial and error strategies. This strategy is exhibited when making small changes in the source code, compile and run, data entry and retrieving the result, assessing the answer and evaluating and comparing the new answer with the previous ones and their corresponding source code changes (Cuba-Ricardo *et al.*, 2014).

In relation to the evaluations made, the cycled is repeated as many times as the student has patience for, new combinations have to be tested, or changes are made during the process.

4. Conclusions

Monitoring the students' behavior, allowed saving the construction process of the solution algorithm in its transcription or coding to a PL of the oriented objects paradigm. This aspect was later contrasted with the interview with the students, which helped reveal the students' notes in the worksheets, as well as the reasoning followed when determining the solution algorithm.

In general, the observation process of the source code construction, in its different evolutionary status, is of greater value for its analysis rather than observing the final program, due to:

- The analysis of the process along with its explanation by the student, promotes the effective valuation of the programming techniques and particularly that of problem solving.
- Along with the interview with the student, his way of thinking can be known.
- It allows determining which data structures or algorithms are more difficult for the student.
- It reveals the most common skills when using the tools that make possible the solving process of programming problems.

5. Future Work

The most important task remaining in our work is porting the software to an open source operating system. This allows the application of such methodology to another context,

and encourages us to search for other results that should improve the dimensions and indicators in each methodology stage.

On the other hand, the intention is to capture more images in less time intervals, and the best solution is to save screenshots as video format instead of images.

The final idea is to determine automatically some behavioral patterns manifested by the students when solving programming problems. This should avoid the need for extensive human review; and should be possible through new software, that receives the video as input and after processing the information it returns the time intervals in which these patterns appear. We must also define these patterns.

References

Cruz Ramírez, M. (2002). *Estrategia Metacognitiva en la Formulación de Problemas para la Enseñanza de la Matemática*. PhD, ISPH "José de la Luz y Caballero", Holguín.

Cuba Ricardo, G., Leyva Figueredo, P.A., Mendoza Tauler, L.L. (2014) Learning strategies of informatics contestants. *Olympiads in Informatics: International Journal*, 8, 35–48. http://www.ioinformatics.org/oi/pdf/v8_2014_35_48.pdf

Deek, F.P, Hiltz, S.R, Kimmel, H., Rotter, N. (1999). Cognitive assessment of students' problem solving and program development skills. *Journal of Engineering Education*, 88(3), 317–326.

Hosseini, R., Vihavainen, A., Brusilovsky, P. (2014) Exploring problem solving paths in a Java programming course. *Psychology of Programming Interest Group Annual Conference.* 65–76.

Kolstad, R., Piele, D. (2007). USA computing olympiad (USACO). *Olympiads in Informatics: International Journal.* 1, 105–111. http://www.mii.lt/olympiads_in_informatics/htm/INFOL016.htm

Maggiolo, S., Mascellani, G., Wehrstedt, L. (2014). CMS: a growing grading system. *Olympiads in Informatics: International Journal*, 8, 123–132. http://www.ioinformatics.org/oi/pdf/v8_2014_123_132.pdf

Mares, M., Blackham, B. (2012). A new contest sandbox. *Olympiads in Informatics: International Journal*, 6, 100–109. http://www.mii.lt/olympiads_in_informatics/htm/INFOL094.htm

Revilla, M. A., Manzoor, S., Liu, R. (2008). Competitive learning in informatics: the UVa online Judge experience. *Olympiads in Informatics: International Journal*, 2, 131–148. http://www.mii.lt/olympiads_in_informatics/htm/INFOL035.htm

Schoenfeld, A.H. (1985) *Mathematical Problem – Solving*. New York, Academic Press.

Surakka, S., Malmi, L. (2004). Cognitive skills of experienced software developer: Delphi study. In: Korhonen A., Malmi, L. (Eds), *Kolin Kolistelut–Koli Calling 2004. Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education.* Finland, Koli, 37–46

Verhoeff, T. (1997). *The Role of Competitions in Education*. Paper presented at the Future World: Educating for the 21st Century. A Conference and Exhibition at IOI' 97. http://olympiads.win.tue.nl/ioi/ioi97/ffutwrld/competit.pdf

Verhoeff, T. (2008). Programming task packages: peach exchange format. *Olympiads in Informatics: International Journal*, 2, 192–207. http://www.mii.lt/olympiads_in_informatics/htm/INFOL019.htm

White, G.L., Sivitanides, M.P. (2002) A theory of the relationships between cognitive requirements of computer programming languages and programmers' cognitive characteristics. *Journal of Information Systems Education*, 13(1), 59–68.

Weinstein, C.E., Underwood, V.L. (1985). Learning strategies: the how of learning. In: Segal, J.W., Chipman, S.F., Glasser, R. (Eds), *Relating Instruction to Research, Volume 1 of Thinking and Learning Skills*. London, Lawrence Erlbaum Associates.

G. Cuba-Ricardo is a doctoral student at the Curricular Doctorate of the University of Pedagogical Sciences of Holguín, Cuba. He is a researcher at the Department of Resource Development for Learning. His main research interest is the role of computer programming in educational processes and contestant training. He is a consultant and a coach of the informatics contests team in Holguín city.

M.T Serrano-Rodríguez is a Specialist in Pedagogy and Psychology, and Bachelor of Education specializing in Chemistry. She presents publications in scientific events as FIMAT, ENFIQUI and Pedagogy, with themes related to ICT and Chemistry. She is a researcher of educational orientation for the use of ICT in the learning process.

A. Leyva-Figueredo holds a Ph.D. in Pedagogical Sciences. He is the Director of the Center of Studies for Labor Education and Coordinator of the Doctorate Collaborative Curricular Program at the University of Pedagogical Sciences of Holguín. He has extensive experience in labor education, doctoral training, research methodology, professional skills and professional guidance. Also, he is a member of the Provincial Scientific Committee of Science, Technology and Environment (CITMA) and a Permanent Member of the Evaluation Board to get the Scientific Degree of Doctor in Pedagogical Sciences.

L.L. Mendoza-Tauler received her Ph.D. in Pedagogical Sciences in 2001. She is the Director of the Center of Studies in Educational Research at the University of Pedagogical Sciences of Holguín and Coordinator of the Ph.D. Program of UBV-2 Caracas, Venezuela. She teaches at the Ph.D., Masters and Qualified Program Studies in Venezuela and Perú. She is a member of the Academy of Sciences of Cuba in the Commission of Social Sciences. She is also a member of the Provincial Scientific Committee of Science, Technology and Environment (CITMA) and of the Evaluation Board to get the Scientific Degree of Doctor in Pedagogical Sciences.