Self-explaining from videos as a methodology for learning programming

Viviane Cristina Oliveira Aureliano

Center of Informatics - Federal University of Pernambuco
Federal Institute of Pernambuco - Campus Belo Jardim
vcoa@cin.ufpe.br

Keywords: POP-I.A. Learning to program; POP-I.B. Choice of methodology; POP-II.A. Novices.

Abstract

The literature has shown that novice learners have several difficulties in acquiring programming skills. As a way to overcome these difficulties, novice learners should be guided in the process of studying the programming process. One evidence-based practice for improving student learning by guiding them while they are studying from some instructional material is through the use of self-explanations. Video recordings have been proposed as an ideal instructional material for presenting the dynamics of programming to novices. In this context, my research proposes that novices should be guided in the process of studying video recordings by means of self-explanation. In order to evaluate the benefits of using the proposed methodology, two studies will be realized as part of this research.

1. Introduction

Years of research have shown that novice learners experience several difficulties in acquiring programming skills. They have fragile knowledge in learning programming (Perkins and Martin 1986). One type of fragile knowledge is the inert knowledge, which is the knowledge novice learners have, but fail to retrieve when it is necessary (CTGV 1992). Novices use inappropriate study methodologies and do not take constant effort to develop their programming skills (Gomes and Mendes 2007). However, the major difficulty experienced by novices is how to combine and properly use the basic structures to build a program (Caspersen & Kölling 2009).

Stepwise Improvement is a conceptual framework that describes programming as an incremental process that encompasses three different activities, extension, refinement and restructuring (Caspersen 2007; Caspersen & Kölling 2009). In the area of programming education, the authors advocate the application of the framework in order to guide the process of teaching and learning programming. In this case, it provides guidance regarding the structure of the instructional materials in a way that novices learn programming by extending, refining and restructuring small pieces of code systematically and incrementally during the course.

Bennedsen and Caspersen (2008) have proposed the use of videos as an ideal medium for revealing the programming process for learners. Instead of textbooks that are static, the authors advocate that videos are an important instrument to expose the programming process since they are essentially a dynamic medium. Caspersen (2007) extended this idea by proposing in his doctoral dissertation that worked examples in videos could be used as an instructional material in an introductory programming course. The application of worked examples and practice exercises that are similar in structure but diverse in situations of use has been demonstrated as a good strategy to promote the transfer of learning (Clark, Nguyen and Sweller 2006).

Self-explanation is a type of dialog that learners have with themselves while they are learning from different instructional materials (Bielaczyc, Pirolli & Brown 1995; Chi, Bassok, Lewis, Reimann & Glaser 1989; Clark et al. 2006). It is an effective learning activity that leads learners to move further in their knowledge about a subject (Fonseca & Chi 2011) by building new knowledge through the refinement of the given information (Chiu & Chi 2014).

As a way to overcome the aforementioned difficulties, novice learners should be guided in the process of studying the programming process. This guidance will be provided by means of the use of self-explanation practice. Since video recordings have been proposed as an ideal instructional material for

exposing the programming process to novices, the self-explanation practice will be applied with this kind of instructional material. In the presented context, my PhD research proposes that novice learners should be guided in the process of studying video recordings by means of self-explanation. This methodology of using videos in conjunction with self-explanations will adopt the programming pathway defined by the Stepwise Improvement framework.

2. Theoretical framework

2.1 Cognitive Load Theory

Cognitive Load Theory (CLT) provides a set of learning principles and related instructional guidelines that are proven to result in instructional environments that are more effective regarding students learning (Clark et al. 2006). In order to provide such a set of principles and guidelines, CLT is closely aligned with the human cognitive structure and the way that people learn. In their book, the authors of the theory explain that the human cognitive structure has schemas or patterns as units of information. In order to manipulate and store this information, the human cognitive structure has also two memory systems, the working memory and the long-term memory. The working memory is the active partner of our cognitive structure. It is responsible for processing new information while we are learning and also for organizing new schemas from the ones that we already have. The working memory works closely together with the long-term memory. While the working memory has a very limited capacity, long-term memory has a huge capacity for storing information and, once we learn, the resultant schemas are stored on it. Because of this, the long-term memory is our main knowledge repository.

According to Clark et al (2006), while students are learning they have to deal with three different types of cognitive load: intrinsic, germane and extraneous. Intrinsic cognitive load results from the complexity of the instructional content. Germane cognitive load is the productive mental effort required for building new schemas and relevant to the learning process. Extraneous cognitive load is defined as the unproductive (or irrelevant) mental effort that does not lead to learning. In order to have efficient learning environments, teachers have to manage intrinsic cognitive load, increase germane cognitive load and reduce extraneous cognitive load. Doing that, they can free the working memory from irrelevant mental effort. Consequently, it is possible to apply that free space for the work demanded to integrate new information into schemas in the long-term memory.

2.2 Stepwise Improvement

Stepwise Improvement is a conceptual framework that describes programming as a systematic and incremental process that encompasses three different activities, extension, refinement and restructuring (Caspersen 2007; Kölling & Caspersen 2009). Extension activity occurs when the specification is expanded in order to cover more use cases. Refinement activity occurs when abstract code is modified towards an executable code in order to implement the current specification. Restructure activity occurs when an improvement of nonfunctional aspects of a solution is made, but this modification does not involve a change in its apparent behavior. These activities are organized in a three-dimensional space that is explored by programmers while they are building programs.

Caspersen and Kölling (2009) advocate the application of Stepwise Improvement by using guided tours rather than random walks. In order to conduct learners' steps in the programming space offered by the framework, the authors state that teachers should be concerned with the right amount of guidance and scaffolding (Collins, Brown & Newman 1989) given during the instruction. It guarantees to balance important aspects of programming education: train learners' programming skills in a proper manner and, at the same time, maintain the cognitive load while students are learning under control.

In that sense, Stepwise Improvement brings a significant contribution to the programming education field by providing guidance in the three activities that it encompasses. Regarding the extension and restructure activities, it provides guidance in the way that the instructional material is organized. Therefore, programming textbooks, assignments, lectures and examples, among other instructional materials, can be structured using the set of steps defined in the framework. For instance, Stepwise

Improvement has been employed in the organization of the Greenfoot's textbook (Kölling 2010) and the problem solving videos of the Joy of code channel on Youtube (Kölling 2012). In addition, regarding the refinement activity, Caspersen (2007) defined in his doctoral dissertation an object-oriented programming process for teaching novices.

2.3 Worked examples

One effective manner of maximizing the germane cognitive load is by using worked examples. Worked examples are sequences of steps defined to demonstrate one particular task performance or to solve a specific problem (Clark et al. 2006). They are especially important for novice learners, once they represent a step-by-step guide necessary to execute a procedure or reach a goal. For this reason, worked examples help novice learners to build new schemas while they are acquiring a new knowledge or skill.

In that sense, worked examples are an effectively evidence-based instructional material. In their book, Clark et al. (2006) demonstrate some advantages of using worked examples. For instance, learners using pairs of worked examples-problems could spend less time studying and still have equivalent learning results rather than learners using all problems as practice exercises. In addition, the authors showed that the application of worked examples and practice exercises that are similar in structure but diverse in situations of use is a good strategy to promote the transfer of learning.

Worked examples have been used in different subjects, such as Physics (Chi et al. 1989), Chemistry (Crippen & Earl 2007) and Programming (Abdul-Rahman & du Boulay 2014; Caspersen 2007), among others. In addition, they can be shown in various formats and modalities, including text (Chi et al. 1989), web (Crippen & Earl 2007), learning environment (Abdul-Rahman & du Boulay 2014) or video recordings (Caspersen 2007).

2.4 Self-explanations

Self-explanation is a kind of dialog that learners have with themselves while they are learning from different instructional materials (Bielaczyc et al. 1995; Chi et al. 1989; Chi et al. 1994; Clark et al. 2006; Crippen & Earl 2007). Fonseca and Chi (2011) presented a literature review about the self-explanation effect (Chi et al. 1989) and its effectiveness as a learning activity that leads people to produce overt output that goes further the presented information. In this sense, the process of self-explaining benefits the building of new knowledge through the refinement of the given information, associating this new information with prior knowledge, reasoning about it and connecting it with other different pieces of information (Chi & Chiu 2014). Self-explanations also promote the exploration of worked examples by learners in a deeper way and maximize the germane cognitive load (Clark et al. 2006). Because of their evidence-based benefits, self-explanations are considered one of the seven practices recommended for improving student learning (Pashler et al. 2007).

The self-explanation effect was demonstrated in different subjects, such as Biology (Chi et al. 1994), Physics (Chi et al. 1989) and Programming (Bielaczyc et al. 1995), among others. In addition, self-explanations can be elicited from various formats and modalities of instructional material, including text (Chi et al. 1994; Chi et al. 1989; Bielaczyc et al. 1995) and worked examples (Clark et al. 2006). The activity of self-explanation may be spontaneously generated (Chi et al. 1989). Also, it can be prompted and trained (Bielaczyc et al. 1995; Crippen & Earl 2007).

3. Statement of thesis

As mentioned before, Stepwise Improvement is a conceptual framework that describes programming as a systematic and incremental process that takes place in a three dimensional space of extension, restructure and refinement activities. In order to move in this space, Caspersen and Kölling (2009) advocate guided tours instead of random walks. Guidance has been provided regarding the manner that the instructional material for programming courses may be presented to students and also the programming activity itself. However, the framework does not provide any guidance regarding the way that students may study and comprehend this instructional material.

In addition, the concept of self-explanation was cited previously as well as its benefits as a learning activity. Self-explanations promotes the building of new knowledge through the refinement of the given information, associating this new information with prior knowledge, reasoning about it and connecting it with other different pieces of information (Chi & Chiu 2014). Explaining to oneself is an effective practice recommended for improving students learning (Pashler et al 2007). In particular, this practice is especially important for novice learners in programming and contributes to their learning while they are studying alone (Bielaczyc et al. 1995).

Nowadays, the use of videos for teaching programming is becoming more and more popular. For instance, numerous videos are available in Youtube channels or in organizations as Khan Academy (2014) and Code.org (2014). However, the use of videos for teaching programming is not a novelty. Some years ago, Bennedsen and Caspersen (2008) showed the benefits of using videos as a tool for revealing the programming process for learners. Instead of textbooks that are static, the authors advocate that videos are an important instrument to expose the programming process since they are essentially a dynamic medium. In addition, using videos students can manipulate these videos - play, forward or rewind - in their own pace and how many times they need.

Stepwise improvement has been employed in the incremental structure of different video collections for learning programming. For example, the structure of problem solving videos of the Joy of code channel on Youtube (Kölling 2012) followed this framework. Also, Caspersen (2007) adopted this idea by proposing in his doctoral dissertation that worked examples in videos could be used as an instructional material in an introductory object-oriented programming course.

I agree with Bennedsen and Caspersen (2008) that videos are an important medium for exposing the programming process. In particular, because of their aforementioned advantages, I believe that videos are an important resource when students are studying programming outside the classroom. Also, I believe that the process of studying programming can be guided by prompting self-explanations from the students while they are studying from these videos. In this context, the main objective of this PhD dissertation is to provide guidance regarding the way students are studying and comprehending videos for learning programming. This guidance will be provided by means of prompting self-explanation while novice learners are studying the programming pathway defined in the Stepwise Improvement framework.

In order to do that, the main research question that guides this PhD is *how novice learners in programming could benefit from using self-explanations while studying video recordings that expose the programming process?* In this sense, novice learners would benefit if they achieve better results, obtain higher grades, code faster or make fewer errors.

4. Research goals

Aiming to answer the question which conducts this research, I am realizing two different studies. Both studies are based on the previous researches developed in the works of Bielaczyc et al. (1995) and Chi et al. (1994), but instead of studying plain texts, learners are using videos as an instructional material.

In the first study, I am using Java as programming language, Greenfoot (2011) as development tool and the problem solving videos of the Joy of code channel on Youtube (Kölling 2012). It has the following objectives: (i) check whether the proposed self-explanations prompts are appropriate for the videos; (ii) analyse when to use the self-explanations prompts and the videos together; (iii) verify if there is any benefit of using self-explanations while studying videos that expose the programming process (if so, which are these benefits); and (iv) refine the self-explanations prompts and the manner that they should be used. In order to do that, the participants in the study should be divided in three different groups: (i) the instructional group in which the self-explanations are asked while showing the video; (ii) the instructional group in which the self-explanations are asked only when the video ends; and (iii) the control group.

This first study consists of three phases: (i) pre-intervention phase; (ii) instructional intervention phase; and (iii) post-intervention phase. The objective of the pre-intervention phase is to provide the participants with a common prior knowledge of the programming language Java and familiarize them

with the self-explanation prompts and the think aloud process. The objective of the instructional intervention phase is training the two instructional groups in how to use the self-explanation prompts. In that sense, for the case of the first instructional group, the video is divided in goals and for each goal the video is paused and the student is prompted with the self-explanation questions. On the other hand, for the second instructional group, the self-explanations questions will be prompted only when the video finishes. At the end of this study, the post-intervention phase aims at collecting data from all participants after the instructional intervention phase.

The students are assessed in two different ways. First, after studying each programming video, students are asked to solve similar problems in order to assess their process of transfer of knowledge. Second, they are assessed by means of the explanations provided by them while studying the programming videos in the instructional and post-intervention phases. In addition, I am getting feedback from the students about their usage of the videos and self-explanation prompts for learning programming.

As mentioned previously, worked examples are an effectively evidence-based instructional material that accelerates the novice's process of learning when adopted together with practice exercises. The concept of worked examples is closely connected with the self-explanations' concept, since self-explanations consist in a manner of promoting deeper learning of worked examples. Aiming to answer the question which conducts this research, I believe that an effective approach is to vary the kind of video recordings exposing the programming process adopted on it. For this reason, the second study will be extremely similar in structure to the first one regarding the objectives, types of group participants, sequence of phases, programming language and development tool. However, it will vary regarding the following characteristics: (i) a larger number of participants; (ii) a refined set of self-explanations prompts and practice problems; and (iii) worked examples in videos as instructional materials instead of using the problem solving videos of the Joy of code channel on Youtube. To do that, I am producing these worked examples in videos in a similar structure proposed in the Joy of code's videos.

5. Dissertation status and expected contributions

I am enrolled in an academic PhD programme in Computer Science at the Informatics Centre of the Federal University of Pernambuco, Brazil. I am also a teacher at the Federal Institute of Pernambuco, Brazil, where I mainly teach programming language subjects. My supervisor is Professor Patrícia Tedesco. The PhD programme in which I participate must be completed in four years and I am at the beginning of the fourth year. I have already completed all six courses that I needed. Also, I defended my research proposal in November 2013. Currently, I am working as a visiting PhD student with Professor Michael Caspersen at Centre for Science Education at Aarhus University, Denmark.

The expected contributions of this PhD research are: (i) a methodology for studying programming from videos that expose the programming process in a deeper manner by self-explaining them; (ii) a study to analyse how novice learners in programming would benefit from using self-explanations while studying videos that expose the programming process; and (iii) a suite of worked examples using Java as programming language and Greenfoot as development tool, enriched with self-explanations prompts. All these contributions use the Stepwise Improvement as the principal conceptual framework.

6. Acknowledgments

I would like to thank CAPES Foundation for the scholarship provided under report nº 1127-91-3.

7. References

Abdul-Rahman, S. S., & du Boulay, B. (2014). Learning programming via worked-examples: Relation of learning styles to cognitive load. *Computers in Human Behavior, 30,* 286-298. doi:10.1016/j.chb.2013.09.007

Bennedsen, J., & Carpersen, M. E. (2008). Exposing the Programming Process. In J. Bennedsen, M. E. Carpersen, & M. Kölling (Eds.), *Reflection on the Theory of Programming: Methods and Implementation* (pp. 6-16). Berlin, Germany: Springer-Verlag.

Bielaczyc, K., Pirolli, P., & Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem solving. *Cognition and Instruction, 13,* 221-253. doi:10.1207/s1532690xci1302_3

Caspersen, M. E. (2007). Educating novices in the skills of programming. PhD dissertation PD-07-4, Department of Computer Science, University of Aarhus.

Caspersen, M. E., & Kölling, M. (2009). STREAM: A First Programming Process, ACM *Transactions on Computing Education, 9 (1),* 4:1-29. doi:10.1145/1513593.1513597

Chiu, J. L., & Chi, M. T. H. (2014). Supporting Self-Explanation in the Classroom. In V.A. Benassi, C.E. Overson, & C.M. Hakala (Eds.). *Applying science of learning in education: Infusing psychological science into the curriculum.* Retrieved from: http://bit.ly/KbYLtG

Chi, M. T. H., Bassok, M., Lewis, M., Reimann, M., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13,* 145-182. doi:10.1207/s15516709cog1302_1

Chi, M. T. H, deLeeuw, N., Chiu, M., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18,* 439-477. doi:10.1016/0364-0213(94)90016-7

Clark, R., Nguyen, F., & Sweller, J. (2006). Efficiency in learning: evidence-based guidelines to manage cognitive load. San Francisco, CA: John Wiley & Sons, Inc.

Cognition and Technology Group at Vanderbilt - CTGV (1992). The Jasper series as an example of anchored instruction: Theory, program description, and assessment data. *Educational Psychologist, 27,* 291-315. doi:10.1207/s15326985ep2703_3

Collins, A., Brown, J. S., & Newman, S. E. (1987). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. Technical Report No. 403, University of Illinois.

Code.org (2014) Retrieved from: http://code.org/learn

Crippen, K. J., & Earl, B. L. (2007). The impact of Web-based worked examples and self-explanation on performance, problem solving, and self-efficacy. *Computers & Education, 49 (3),* 809-821. doi:10.1016/j.compedu.2005.11.018

Gomes, A., & Mendes, A. J. (2007) Learning to Program - Difficulties and Solutions. In *Proceedings of the International Conference on Engineering Education, Coimbra, Portugal.* Retrieved from: http://bit.ly/1ldTqkE.

Greenfoot (Version 2.3.0) [Computer software] (2011) Retrieved from http://www.greenfoot.org/door

Fonseca, B. & Chi, M.T.H. (2011). Instruction based on self-explanation. In R. E. Mayer, & P. A. Alexander (Eds.), *The Handbook of Research on Learning and Instruction* (pp. 296-321). NY: Routledge Taylor and Frances Group.

Khan Academy (2014) Retrieved from https://www.khanacademy.org/

Kölling, M. (2010) Introdution to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. New Jersey, USA: Pearson Higher Education.

Kölling, M. (2012). *The Joy of code* [Video recordings]. Retrieved from http://bit.ly/1jbTbHq

Pashler, H., Bain, P., Bottge, B., Graesser, A., Koedinger, K., McDaniel, M., and Metcalfe, J. (2007). Organizing Instruction and Study to Improve Student Learning (NCER 2007-2004). Retrieved from: http://ncer.ed.gov

Perkins, D. N.; Martin, F. (1986) Fragile knowledge and neglected strategies in novice programmers. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers, First Workshop* (pp. 213–229). Norwood, NJ: Ablex.