

Smart Program Visualization Technologies: Planning a Next Step

Roman Bednarik, Andrés Moreno, Niko Myller and Erkki Sutinen

Dept. of Computer Science, University of Joensuu, P.O. Box 111, FI-80101 Joensuu, Finland.

firstname.lastname@cs.joensuu.fi

Abstract

Learning to program is a difficult and complex process that needs to be aided by proper educational tools. The crucial question is if the tool can support the learning or not. The potentials of Program Visualization (PV) tools, especially essential in novice programmers training, were shown in the past. Unfortunately, they are still underutilized and the results of their use are inconclusive. Moreover, the approach of creating general-purpose tools for a general-user is no longer bearable. The tools should be smart and accommodate to the changing needs, goals, and context of the users. This can increase the efficiency, acceptance and usage of PV tools. We perform a critical analysis of the current state-of-practice in PV and smart technologies and propose a taxonomy linking these research tracks. In addition, we present directions for the future of the research in smart program visualization tools.

1. Introduction

No learning process is a straight path. Undoubtedly, everyone has a specific approach to learning. In educational research, this concept has been referred to as a learning style. It is believed that when the preferred approach to learning is supported, the effectiveness of the learning is increased. In classrooms, teachers are supposed to take care of the diverse learners. However, that is not always possible, especially in large groups. Furthermore, the situation is totally different when learners are in isolation. Educational technology researchers have been building tools to support learners and remedy the lack of teacher, but normally this has happened in a uniform fashion. For learners, it is necessary not to be limited to a predefined path, but the environments should be aware of the tasks, abilities, and contexts of the users.

In recent years, the interests have indeed shifted towards the tools that can take the learner into consideration. Particularly, a technique known as

adaptation [5] is one of the most promising ones. Adaptation refers to the means of supporting the user within different and varying settings, backgrounds and cultures, to further exploit the available knowledge and improving technology. Many personalized and intelligent learning environments have been created employing some sort of adaptation. However, it seems that these systems still possess several limitations and are only targeted to specific content areas. Moreover, the adaptation technique is especially used to improve learning materials and their delivery. It is still an open question whether similar techniques can be considered when enhancing learning tools to support learning.

When considering the effectiveness of software visualization (SV) for educational purposes, the research results have been inconclusive [10]. One of the possible causes could be the lack of support for diverse learners in different learning contexts.

Our intention is to investigate the possibilities of enhancing PV tools by smart techniques to support and increase the learning of programming. We argue that only applying the existing adaptation techniques to learning tools is not enough because the learning materials and tools have distinct focus. Thus, we need to examine the nature of learning tools and their relation to the learning process. From this analysis we will then build our view on smart learning tools that support the whole learning process.

In Section 2, we introduce our view on learning process and its relation to the learning tools. In Section 3, we review research in the areas of program visualization and smart techniques. The current situation in program visualization tools and how they could be adapted is discussed in Sections 4 and 5, respectively. The future work and discussion is presented in Section 6.

2. What Is Needed from a Learning Tool

Our view on the interplay between learning process and the tool to support it consists of at least four components affecting it. Firstly, it is necessary to know

what should be learned. Unintentional learning can also happen, although it is hard to support it with tools. The content of learning should be somehow explained or expanded by the tool. This will allow the learner to focus on the part that needs attention. Secondly, all learners have personal preferences for the style *how* the learning is organized. Learners should be able to navigate through the content in a way suitable for them and the tool should support this navigation. Thirdly, during the whole learning process, *interaction* in the forms of responses, support and feedback are crucial. This interaction should guide the learning process. If a learner has problems during the process, s/he should be able to ask for help. Furthermore, the feedback should help the learner to set up new learning goals and enable reflection on the work done. Finally, the learning happens always in some *context* of culture, background and settings. The learning tools should always take into consideration what the context is and accordingly accommodate an appropriate support for it. We argue that for a learning tool to be maximally effective, it has to support every part of the learning process as presented above. In other words, if a tool does not consider sufficiently some of the parts of the learning process, it will most likely fail when used in learning.

3. Literature Review

3.1. Smart Techniques

Several techniques have been previously used in educational tools to achieve a smart support for learning processes. Adaptive learning environments have been a popular field of study in the recent years [5]. Adaptation refers to the adjustments in an educational environment aiming to (1) accommodate learners' needs, goals, abilities, and knowledge, (2) provide appropriate interaction, and (3) personalize the content [1]. However, besides the users' characteristics, also usage (interaction) data and environment data shall be included when modeling the user [11]. Adaptation to the environment of the user comprises the context of use, platform, and location.

According to Beaumont and Brusilovsky [1], in the context of Adaptive Hypermedia (AH) two types of adaptation are possible: *adaptive presentation of the content* and *adaptive navigation support*. In content adaptation, the materials are modified to suit the user according to the user model. The adaptive navigation support means that the different directions the user can take are limited or guidance is given in the selection of the direction. With an increasing research in this area,

this taxonomy has been enriched [4][5]. Namely, *adaptation of modality* is such a development in the branch of adaptive content presentation where the modality of the content is changed, for example to accommodate the learning style of the student. In adaptive navigation support, advances have been done, for instance, towards adaptive link generation in which new links between documents are added to the system automatically and they are recommended with dynamically changing criterion.

Besides adaptivity, a form of smart support can be achieved by using *agent technology* [8]. The autonomous software agents can analyze some part of the learning process and give support when they sense that an aid or response is needed or when the user asks for help. Agents can be used in several contexts. They can be seen as a way to collect data for user modeling both by observing and by asking the user when needed. Furthermore, when there is no peer available for a conversation, social agents can try to interact with the user to further support the learning process.

3.2. Program Visualization

Program-visualization (PV) systems concentrate on explaining the execution of computer programs, providing an access to dynamic and often hidden processes during program run-time. Another field of research close to PV is Algorithm Visualization (AV). AV tools range from predefined algorithm animations and static pictures to environments enabling the learners to design their own AVs. Both fields, PV and AV, are part of the broader area of Software Visualization (SV), which includes all the systems that convey information about software in a graphical way.

PV tools have approached the communication of program execution process in different ways. BlueJ [13] is the most successful tool developed for novices learning object-oriented programming. It lets the users to interact with classes by creating objects and calling their methods through a visual interface. Other PV systems visualize the programs during their execution. *Jeliot 3* [15] shows the execution step-by-step through animation, for instance, the evaluation of a condition or an expression is shown in details. *JeCo* [16] is a collaborative extension of Jeliot 3 where users can chat with each other and to post program code and messages to a collaboration view. *Javavis* [17] displays the state of the program and the messages sent by objects with UML-like notation.

Although there is a great potential in the PV or SV in general, the results of evaluating SV tools are inconclusive in terms of if and how their use actually

contributes to achieving better learning results. In a meta-study of AV evaluations, Hundhausen et al. [10] concluded that instead of concentrating on *what and how* to display, the focus should be on studies of the *ways how AV systems are used*. It also reveals that AV tools are more engaging when combined with common learning tasks, such as prediction exercises.

A study reported in [12] shows how the use and evaluation of AV tools does not reflect the full potential of the tools. The claim is that most of the past evaluations were carried out in constrained environments, for example, in time limited tests, and that those tools would perform better in open environments. These results could also lead to the conclusion that current tools could perform better, if they were able to adapt to the learning context. Thus, the problem of inconclusive empirical results might lie in the fact that the tools are often designed in a one-fits-all fashion, and the problem can materialize when a tool is used by a diverse population of learners.

4. Program Visualization for Learning

One of the solutions to the problem presented in the previous section is to employ adaptation. Although several program visualization tools exist, there is very limited support for adaptivity and adaptability in the PV systems. These, often novice-oriented, tools fail to support learning in many respects: They (1) do not provide any means to change the actions and content to accommodate the learning process of the student, (2) do not respect the changing context of learning and (3) do not proactively interact to provide guidance. The tools shall anticipate progress in learning, as it is an expected goal of their use, and they should act upon it.

There are a few systems that utilize different techniques when adapting to the needs of users. However, the systems are limited in the scope of their use and visualization. *Seal* [14] uses social agents to support the viewing of the program visualizations. The social agents negotiate with the user how the visualization proceeds and if the results of the visualization are as expected. Moreover, in problem situations the agent tries to explain what went wrong. *WADEIn* [7] adapts the level of details in the visualization to the knowledge of the learner when visualizing the expression-evaluation. The learners' understanding in terms of elementary knowledge elements is monitored and the information is stored into a centralized user model repository. The adaptive system adjusts the steps in the animation of an algorithm to focus on elements that are not yet understood as well as the other parts.

A different approach is taken in *Jeliot I* [19] where the users can adapt the visualizations to fit better into their mental models. This can be achieved by changing the visualization parameters of the visual objects. For example, the shape, color, and style used in the visualization can be adjusted. In that way, the program visualization becomes *adaptable*.

5. Making Visualization Tools Smart

In these sections, we suggest ways how the PV tools could be enriched to support all components of the learning process better. Approaching the future of adaptive PV, it is reasonable to consider how the present PV systems could be enriched by and benefit from the current smart techniques.

5.1. Smart Presentation of Content in PV

Considering the adaptive presentation of content, it is mostly the "*what*" component of learning process which a smart PV system would operate on and support. An obvious solution would be to provide different modality of the visualization to match learners' cognitive styles by text, sound, animation etc. For instance, when the principles of conditional branching are explained, some learners might prefer a textual information while others animation. Smart PV tools shall strive to match the modality of the presentation to the one preferred by the learner.

In the area of educational AH, the idea of inserting and removing some fragments of text is often used. Inspired by this approach, we propose to adapt the visualization by inserting and removing fragments of representations of programs during the visualization. For example, BlueJ and other visual design tools could adaptively hide certain class information and display other with additional explanations. Systems similar to Jeliot could show animations of programming constructs that are relevant to the students' current learning goals and hide unwanted constructs. Moreover, some of the PV systems can provide different views of the same concept. For example, in Jeliot together with a data structure view, a call tree of program execution is shown to the user. Smart PV tools should provide suitable views to different users in changing stages of the learning process.

5.2. Smart Navigation support in PV

Applying techniques similar to adaptive navigation support, in our opinion, is not that straightforward. We see the adaptive navigation support as allowing the

sequences of steps in a learning process to be altered from a fixed model. In other words, these techniques could be used to support the “*how*” component of a smart PV. While learning a programming concept, the steps to achieve a maximal understanding might differ from person to person.

There are several possible directions how the adaptive navigation support could be implemented in PV tools. We propose to implement them by adding a supportive set of examples and problems. Techniques from AH systems, like sorting of or link generation between materials can be applied to let the students choose the next example from the relevant ones, or to solve a problem related to the learned concept. In the systems, such as JeCo, new links between messages and materials could be generated.

5.3. Proactive Interaction in PV

Autonomous agents could help the learning process in terms of proactive *interaction*, feedback, and guidance. A proactive component of a smart PV tool could appropriately combine guidance and freedom to change the intensity of proactive intervention, depending on the cognitive style of the learner: some learners might prefer independent learning while others appreciate assistance. For instance, in the program comprehension task they could ask questions that concentrate the learner’s attention on certain aspects of the program depending on the program and abilities of the learner. In collaborative settings, the tool could propose collaboration partners who have knowledge about the concepts the students are learning.

5.4. Smart Context-Awareness in PV

Some of the current AH frameworks already allow for a centralized user modeling, material viewing and invocation of external tools. For instance, the *Knowledge Tree* [6] is a web-based user-modeling portal that allows extensions through new components. We see a possibility to extend such adaptive learning systems to cooperate with PV tools. This way, PV tools can benefit from already implemented techniques and modeled users. The adaptive learning systems can retrieve more information about the learners through the interaction with PV tools, updating the user model more accurately, leading to synergy.

6. Discussion and Future work

Programming is a complex domain in which learners need a constructive support from their learning environments. We present an argument that, similarly as in other learning domains, the modern PV tools cannot be designed and implemented without paying attention to the most important components of the learning process: the learners, their tasks and context where learning happens. The current PV tools lack of recognizing these needs: the tools often animate the program automatically; the animation is designed according to experts’ mental models and based on unfamiliar constructs; and the context of learning is not considered as well as learners’ cognitive styles and abilities. Not only this approach is opposite to the constructivist view on learning [3], but also the real effects of the use are often lower than expected [10].

As a solution to the problem, we propose to involve some of the smart techniques reviewed in previous sections. The research in educational AH has intensively advanced towards smarter systems during last decade [5]. However, its implications to PV technologies are rare. We believe that adaptation and agent technologies could increase learners’ engagement, and provide them with deeper understanding and meaningful learning. As a result, the problems connected with the utilization of the PV tools could be solved. We propose that the first step of enriching PV tools by adaptive methods could be made by coupling PV tools with existing adaptive frameworks and mechanisms.

However, it is necessary that a sound methodological framework is developed to evaluate properly the proposed solution. Although the previous research in AH is rich and inspiring, our purpose is not to blindly adapt all the methods developed in past. This is not even possible as AH is concerned with the delivery of materials while the learning tools with the information processing. Rather, the effects of adaptation shall be thoroughly examined. For that reason, we argue that the future research in smart PV tools shall be focused on developing the evaluation. A repertoire consisting of controlled laboratory studies, usability tests, as well as longitudinal, observational studies, remote logging, interviews and real teaching experiments shall be involved. The evaluation of the outcomes of future PV tools application shall not only consider the actual knowledge gathered through learning, but it also shall measure the produced mental models [9] and the levels of engagement.

Advances in available technologies shall be used to monitor the learners in order to foster the creation of accurate user models. Such technologies as eye-movement tracking could be involved to estimate the

focus of attention on certain elements of animated program, or to infer the current cognitive load induced by the animation. Such systems are currently used in laboratory-based evaluations and provide insights into how different parts of the interface are utilized by different users during program animation [2].

Other challenges of the research in the future of advanced PV tools include the issues of (1) efficient integration into programming courses and (2) support for different sub-areas of programming such as comprehension, debugging, or creating programs from a scratch or template. If the PV tools are included into the computer science curricula, it is necessary to integrate proper pedagogical models into them. Furthermore, the tools should be aware of the underlying course structure and cooperate with course-management tools. Another form of adaptation then arises: future PV tools should adapt to the changing institutional and course needs.

7. References

- [1] I. Beaumont, and P. Brusilovsky, "Adaptive educational hypermedia: From ideas to real systems", H. Maurer (Eds.), *Proc. of ED-MEDIA'95*, AACE, 1995, pp. 93–98
- [2] R. Bednarik, A. Moreno, N. Myller, E. Sutinen, E., and M. Tukiainen, "Effects of the Experience on the Gaze Behavior during Program Animation", Accepted to the *17th Annual Workshop of Psychology of Programming Interest Group*, Brighton, UK, June 28–July 1, 2005.
- [3] M. Ben-Ari, "Constructivism in Computer Science Education", *J. of Computers in Mathematics & Science Teaching*, 20 (1), 2001, pp. 45–73.
- [4] P. Brusilovsky, "Methods and techniques of adaptive hypermedia", *User Modeling and User-Adapted Interaction*, 6 (2/3), 1996, pp. 87–129.
- [5] P. Brusilovsky, "Adaptive hypermedia", *User Modeling and User Adapted Interaction*, 11 (1/2), 2001, pp. 87–110.
- [6] P. Brusilovsky, and H. Nijhawan, "A Framework for Adaptive E-Learning Based on Distributed Re-usable Learning Activities", M. Driscoll and T. C. Reeves (Eds.), *Proc. of World Conf. on E-Learning*, AACE, 2002, pp. 154–161.
- [7] P. Brusilovsky, and H. Su, "Adaptive Visualization Component of a Distributed Web-Based Adaptive Educational System", S.A. Cerri and G. Gouardères and F. Paraguaçu (Eds.), *Intelligent Tutoring Systems*, Vol. 2363 of LNCS, Springer-Verlag, 2002, pp. 229–238.
- [8] S. Franklin and A. Graesser, "Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents", *Proc. of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, Vol. 1193 of LNCS, Springer-Verlag, 1996, pp. 21–35.
- [9] J. Good, and P. Brna, "Program comprehension and authentic measurement: a scheme for analysing descriptions of programs", *Int'l J. of Human-Computer Studies*, 61 (2), 2004, pp. 169–185.
- [10] C.D. Hundhausen, S.A. Douglas, and J.T. Stasko, "A Meta-Study of Algorithm Visualization Effectiveness", *J. of Visual Languages & Computing*, 13 (3), 2002, pp. 259–290.
- [11] A. Kobsa, J. Koenemann, and W. Pohl, "*Personalized hypermedia presentation techniques for improving online customer relationships*", Tech. report No. 66 GMD, German National Research Center for Information Technology, St. Augustin, Germany, 1999.
- [12] C. Kehoe, J. Stasko, and A. Taylor, "Rethinking the evaluation of algorithm animations as learning aids: an observational study", *Int'l J. on Human-Computer Studies*, 54 (2), 2001, pp. 265–284.
- [13] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg, "The BlueJ system and its pedagogy", *J. of Computer Science Education*, 13 (4), 2003.
- [14] R. MirafTabi, "Intelligent Agents in Program Visualizations: A Case Study with Seal", *Proc. of the First Int'l Program Visualization Workshop*, Porvoo, Finland, July 2001, pp. 53–58.
- [15] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, "Visualizing Programs with Jeliot 3", *Proc. of Advanced Visual Interfaces*, 2004, pp. 373–376.
- [16] A. Moreno, N. Myller, and E. Sutinen, "JeCo, a Collaborative Learning Tool for Programming", *Proc. of IEEE Symposium on Visual Languages and Human-Centric Computing*, Rome, Italy, 2004. pp. 261–263.
- [17] R. Oechsle, T. Schmitt, "JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI)", Diehl, S. (Ed.), *Software Visualization*. Vol. 2269 of LNCS, Springer-Verlag, 2002, pp. 176–190.
- [18] K.A. Papanikolaou, M. Grigoriadou, H. Kornilakis, and G.D. Magoulas, "Personalising the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE", *User-Modeling and User-Adapted Interaction*, 13 (3), 2003, pp. 213–267.
- [19] E. Sutinen, J. Tarhio, T. Teräsvirta, "Easy Algorithm Animation on the Web", *Multimedia Tools and Applications*, 19 (2), 2003, pp. 179–184.