HUMAN TECHNOLOGY

An Interdisciplinary Journal on Humans in ICT Environments ISSN: 1795-6889

www.humantechnology.jyu.fi Volume 4 (1), May 2008, 4–8

Guest Editor's Introduction

PSYCHOLOGY OF PROGRAMMING: LOOKING INTO PROGRAMMERS' HEADS

Jorma Sajaniemi
Department of Computer Science and Statistics
University of Joensuu, Finland

Psychology of programming (PoP) is an interdisciplinary area that covers research into computer programmers' cognition; tools and methods for programming related activities; and programming education. The origins of PoP date back to late 1970s and early 1980s, when researchers realized that programming tools and technologies should not be evaluated based on their computational power only, but also on their usability from the human point of view, that is, based on their cognitive effects. The hope of such a new approach was that programmers would make fewer errors, produce better software, and work more efficiently. In the first Workshop on Empirical Studies of Programmers, Ben Shneiderman listed "several important destinations for researchers: refining the use of current languages, improving present and future languages, developing special purpose languages, and improving tools and methods" (Shneiderman, 1986, p. 1). During the past two decades, the flow of new languages, tools, and methods has increased rapidly, the scope of programming work has expanded, and research interests have extended to cover group activities. Yet the main goal of PoP—to assist programmers through the benefits of cognitive research—has remained.

The PoP research community consists of cognitive psychologists and computer scientists. The main motivation for computer scientists is the improvement of current tools and the development of new ones, as well as the discovery of general principles concerning humans in the context of programming tasks. Psychologists are interested in new theories of human cognition applicable in other domains too. For them programming—a highly complicated task—provides good opportunities to study high-level cognitive processes in a complex setting. This dual character of PoP manifests itself also in the skills required from researchers: a good knowledge of both cognitive psychology and programming or, better still, psychology, social sciences, and software engineering.

On the other hand, PoP research results are not necessarily limited to the programming domain, but can be applied in other domains that involve design activities in a formal environment. As an example, consider cognitive dimensions (CDs), which were introduced by Green (1989) to describe, compare and control how programming language features affect program design strategies. The dimension role-expressiveness, for example, relates to how well a piece of program code (e.g., "+") reveals its meaning without a need to study the context

URN:NBNfi:jyu-200804151349

of the piece (addition, string catenation, etc.). Later, CDs were developed further and applied to many types of cognitive artifacts, such as educational theorem provers (Kadoda, Stone, & Diaper, 1999), prototyping techniques (Dearden, Siddiqi, & Naghsh, 2003), and music notations (Blackwell & Green, 2003).

Even though the area of PoP seems to be quite narrow—computer programming—it covers a large variety of phenomena, from novices' problems to experts' tacit knowledge, from program design to testing and maintenance, and from short individual programs to huge software systems. Consequently, research methods vary as well. Most often, research methods have been adopted from cognitive psychology (e.g., controlled experiments run in laboratory settings) or social sciences (e.g., field studies with qualitative analysis techniques), but it seems that in many subareas appropriate research methods are yet to be discovered. As many researchers are also computer science educators, they have instant access to novices and, therefore, studies on novices' problems and programming education are frequent. A new source of research materials is provided by various open source communities that make their program code, change logs, and discussions among program developers freely available on the net. These materials represent expert programming in state-of-the-art contexts.

During the past two decades, two important workshop series have been fully devoted to PoP: the Workshop on Empirical Studies of Programmers (ESP), based primarily in the USA, and the Psychology of Programming Interest Group Workshop (PPIG), having a European character. The first ESP workshop was held in 1986 in Washington, DC, the eighth and last one in 1999. Later, this workshop series was incorporated into the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), which, however, has a broader scope than pure PoP and includes implementation aspects and the like. The European conference series, PPIG, started in 1989, and continues to be organized annually. It is more informal in nature than ESP; in addition to fully developed research papers, PPIG proceedings include position papers and suggestions for individual studies. Many of the best papers have later been published in more formal conferences and journals.

The organization behind PPIG workshops, the Psychology of Programming Interest Group, was established in 1987 and—just like the workshop—is informal in nature. For instance, there is no formal committee: Decisions are discussed in an open business meeting held during every workshop. The interest group publishes an electronic newsletter and hosts two mailing lists, a low-volume announcements list plus another list for discussions. In essence, the interest group is an informal collection of people who are enthusiastic about psychological aspects of programming and software engineering[1].

The latest PPIG workshop was held in Joensuu, Finland in July 2007. The scientific program consisted of four half- or full-day tutorials, a doctoral consortium, two keynote addresses, 18 technical presentations, and two discussion sessions. All paper submissions were reviewed by at least two—usually three—anonymous reviewers and papers were accepted in two categories: Full Papers and Work in Progress Reports, as decided by the Program Committee. This special issue of *Human Technology* contains five of those papers, selected based on the reviewers' statements. The papers were re-reviewed and improved for publication in this journal. These papers demonstrate the variability in themes and research methodologies of PPIG workshops.

The first two papers deal with research methodology. In the article "A Coding Scheme Development Methodology Using Grounded Theory for Qualitative Analysis of Pair Programming," Stephan Salinger, Laura Plonka, and Lutz Prechelt consider the analysis of

rich video data that is typical for programming protocols. They have used grounded theory (Strauss & Corbin, 1990), in which the whole coding is based totally on protocol data, and developed a specific coding scheme to be used in the context of pair programming. The article provides guidance for the use of grounded theory in the analysis of rich protocol data when the purpose of a study is to understand cognitive phenomena within a design process. The principles described in the paper apply to domains outside programming, as well.

Rozilawati Razali, Colin Snook, Michael Poppleton, and Paul Garrat have used two methods to evaluate the usability of a semiformal notation that combines UML (Object Management Group, 2007) with B (Abrial, 1996), the latter being a formal notation for describing semantics. The evaluation methods include CDs and the results were analyzed using grounded theory. This paper, "Usability Assessment of a UML-based Formal Modeling Method Using a Cognitive Dimensions Framework," thus demonstrates how one can use several research methods for the usability analysis of tools within formal domains that involve design activities.

The next two papers concentrate on specific details within programming. Sue Jones and Gary Burnett tackle a popular problem: how to predict students' success in learning programming. Earlier work on this area has looked at correlation between programming success and some other property, for example, field dependence (e.g., Mancy & Reid, 2004), inclination to systematic behavior (e.g., Dehnadi, 2006), or self-efficacy (e.g., Wiedenbeck, LaBelle, & Kain, 2004). Jones and Burnett study spatial ability and find a positive correlation between mental rotation ability and programming success in their paper "Spatial Ability and Learning to Program."

Juha Sorva looks at variable-oriented programming paradigm (Sajaniemi & Niemeläinen, 1989) and combines it with the notion of roles of variables (Sajaniemi, 2002). This results in a data-flow description of programs that explicitly classifies variables using a fixed set of categories found in expert programmers' tacit knowledge (Sajaniemi & Navarro Prieto, 2005). The article, "A Roles-Based Approach to Variable-Oriented Programming," also demonstrates how the new notation can be used for mental exercises even without a fully functional implementation.

The final paper, "From Procedures to Objects: A Research Agenda for the Psychology of Object-Oriented Programming Education" by Jorma Sajaniemi and Marja Kuittinen, presents an overview of PoP research in novice education and debates whether existing research literature, which deals mostly with procedural programming, can be applied to current educational practice that is based on object-oriented programming (de Raadt, Watson, & Toleman, 2002). The authors point out fundamental differences that make the use of existing research results in the current context dubious and suggest areas that should be studied if programming education is to be based on research results rather than intuition.

The five papers included in this special issue of *Human Technology* represent studies in research methodology and in small scale programming. Programming in the large, that is, production of complex software systems, is not represented in this set. The reason is simple: There were very few papers on that area in the 2007 PPIG workshop. This is also typical for PoP research in general. Research into the construction of large systems, although highly important, is very expensive and industry partners willing to use their time for such research are hard to find.

There is still a long way to go before PoP can provide an extensive picture of programming and software engineering in general.

ENDNOTE

1. For more information on the Psychology of Programming Interest Group, see http://www.ppig.org

REFERENCES

Abrial, J. R. (1996). *The B-Method: Assigning programs to meanings*. Cambridge, UK: Cambridge University Press.

Blackwell, A., & Green, T. (2003). Notational systems: The cognitive dimensions of notations framework. In J. M. Carroll (Ed.), *HCI models, theories, and frameworks: Toward a multidisciplinary science* (pp. 103–133). San Francisco: Morgan Kaufmann Publishers.

Dearden, A., Siddiqi, J., & Naghsh, A. (2003, April). *Using cognitive dimensions to compare prototyping techniques*. Paper presented at the 15th Annual Workshop of the Psychology of Programming Interest Group, Keele, UK.

Dehnadi, S. (2006). Testing programming aptitude. In P. Romero, J. Good, E. A. Chaparro, & S. Bryant (Eds.), *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group* (PPIG '06; pp. 22–37). Brighton, UK: University of Sussex.

de Raadt, M., Watson, R., & Toleman, M. (2002). Language trends in introductory programming courses. In E. Cohen & E. Boyd (Eds.), *Proceedings of Informing Science and IT Education Conference* (InSITE '02; pp. 329–337). Santa Rosa, CA, USA: Informing Science Institute.

Green, T. R. G. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay (Eds.), *People and Computers V* (pp. 443–460). Cambridge, UK: Cambridge University Press.

Kadoda, G., Stone, R., & Diaper, D. (1999, January). *Desirable features of educational theorem provers: A cognitive dimensions viewpoint*. Paper presented at the 11th Annual Workshop of the Psychology of Programming Interest Group, Leeds, UK.

Mancy, R., & Reid, N. (2004). Aspects of cognitive style and programming. In E. Dunican & T. Green (Eds.), *Proceedings of the Sixteenth Annual Workshop of the Psychology of Programming Interest Group* (PPIG '04; pp. 1–9). Carlow, Ireland: Institute of Technology.

Object Management Group (2007). *Introduction to OMG's Unified Modeling Language (UML)*. Retrieved April 11, 2008, from http://www.omg.org/gettingstarted/what_is_uml.htm

Sajaniemi, J. (2002). Visualizing roles of variables to novice programmers. In J. Kuljis, L. Baldwin, & R. Scoble (Eds.), *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group* (PPIG '02; pp. 111–127). Uxbridge, UK: Brunel University.

Sajaniemi, J., & Navarro Prieto, R. (2005). Roles of variables in experts' programming knowledge. In P. Romero, J. Good, S. Bryant, & E. A. Chaparro (Eds.), *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group* (pp. 145–159). Brighton, UK: University of Sussex.

Sajaniemi, J., & Niemeläinen, A. (1989). Program editing based on variable plans: A cognitive approach to program manipulation. In *Proceedings of the Third International Conference on Human-Computer Interaction on Designing and Using Human-Computer Interfaces and Knowledge Based Systems* (2nd ed.; pp. 66–73). New York: Elsevier Science Inc.

Shneiderman, B. (1986). Empirical studies of programmers: The territory, paths, and destinations. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 1–12). Norwood, NJ, USA: Ablex Publishing Co.

Strauss, A., & Corbin, J. (1990). *Basics of qualitative research: Grounded theory procedures and techniques*. London: Sage Publications, Inc.

Wiedenbeck, S., LaBelle, D., & Kain, V. N. R. (2004). Factors affecting course outcomes in introductory programming. In E. Dunican & T. Green (Eds.), *Proceedings of the Sixteenth Annual Workshop of the Psychology of Programming Interest Group* (PPIG '04; pp. 97–110). Carlow, Ireland: Institute of Technology.

Author's Note

I am grateful to Pablo Romero who organized the reviewing process of the paper that I have coauthored.

All correspondence should be addressed to:
Jorma Sajaniemi
University of Joensuu
P.O.Box 111
FI-80101 Joensuu
Finland
saja@cs.joensuu.fi

Human Technology: An Interdisciplinary Journal on Humans in ICT Environments
ISSN 1795-6889
www.humantechnology.jyu.fi

Considering an MA in Psychology? Learn what kinds of programs will help you reach your goals and check out our list of the best psychology master's programs.Â In this guide, we look at what kinds of psychology master's programs are out there and what the point of these programs is in terms of how they help you build a career. Moreover, we'll present you with a list of the best psychology master's programs for various kinds of psychology. What's the Point of a Master's in Psychology? What is the overall purpose of entering a psychology master's program and earning an advanced degree in psychology? For one, many people choose to get a master's degree in psychology to learn more about a specific subfield or type of psychology. Topics include egoless programming, intelligence, psychological measurement, personality factors, motivation, training, social problems on large projects, problem-solving ability, programming language design, team formation, the programming environment, and much more.Â On the other hand, I've trained thousands of programmers and team leaders, and consulted on hundreds of software projects. I've done more code reviewing, designing, design reviewing, developing requirements, and reviewing requirements. And, I've especially spent a lot of time training would-be software managers, and consulting with them.Â Looking at the books I've written since Psychology, I can now clearly see that I was filling the holes. Taking the books chronologically The problem of learning programming language has existed for a long time and researchers seek to solve this problem. Most instructors agree that there is a problem when teaching programming and many students are unable to understand programming logic. So there must be a method to encourage them. Students need to be motivated to practise study and exchange ideas. Gamification is used as a tool to motivate students and increase their engagement. This paper surveys empirical studies which tackled gamification to encourage computer science students and help them in learning coding or improving their coding skills. The paper will show the results of using this approach with computer science students at university level. Judge programming language by breaking writing down into steps, and look at how much knowledge required to choose between options at each step. Novel language, observe four paper writers working through problems. Example of a macro feature where the correct choice is not clear.Â No body. The evaluation of TED, a techniques editor for Prolog programming. 32 students. 10 week course.Â Developed a language based on review of past psychology of programming results. Experimented with different forumulations of boolean queries. Found confusion over 'AND', precedence/grouping and users totally ignoring parens. The effect of programming language on error rates of novice programmers. GRAIL vs LOGO. 26 1st year undergrads, no experience.