frontiers in Psychology

OPINION
published: 14 September 2016
doi: 10.3389/fpsyg.2016.01390

# Learning from the Past–The Need for Empirical Evidence on the Transfer Effects of Computer Programming Skills

*Ronny Scherer* *

*Faculty of Educational Sciences, Centre for Educational Measurement, University of Oslo, Oslo, Norway*

**Keywords: computational thinking, computer programming, creativity, problem solving, transfer of learning**

## INTRODUCTION TO THE PROBLEM

In recent years, education has put considerable emphasis on the development of twenty-first century skills—a set of skills that can almost universally be applied to a broad range of domains and problems, and that help students to deal with the challenges and demands of complex, real-world problem situations (Pellegrino and Hilton, 2012). Among others, these skills comprise problem solving, creativity, critical thinking, collaboration, adaptability, digital literacy, and computational thinking, and are considered to be critical in our information- and knowledge-rich society (Binkley et al., 2012; Wagner, 2012; Scherer, 2015; Care and Anderson, 2016). Against this background, it has become the designated aim of educators to help students to develop these skills (Kay and Greenhill, 2011). The question of *how* the development of these skills and the ability to transfer them to different contexts and knowledge domains can be fostered has therefore gained significance (Greiff et al., 2014). Nonetheless, this question is by no means trivial, because the transfer of knowledge and skills does not automatically happen, as Tricot and Sweller (2013) argued.

In the pursuit of finding ways to foster twenty-first century skills and their transfer, voices have become loud inspiring education to incorporate computer programming into K-12 curricula (Lye and Koh, 2014). The reactions on these voices have been tremendous; some countries developed an entire curriculum around computer programming (Sturman and Sizmur, 2011; Webb et al., 2016). Behind this development is the belief that fostering programming skills improves students' performance on other critical skills such as creativity and problem solving (Liao and Bright, 1991; Clements, 1995). Mitchel Resnick, the director of MIT's Media Lab and facilitator of the Scratch® programming language, argued that "programming supports "computational thinking," helping you learn important problem-solving and design strategies [...] that carry over to nonprogramming domains" (Resnick et al., 2009, p. 62). Along the same lines, Barr and Stephenson (2011) proposed that computer programming "is a problem solving methodology that can be automated and transferred and applied across subjects" (p. 51). Brown and Kölling (2012) took this argument even further and claimed that the "use of programming skills can allow for a deeper and more direct understanding of the subjects under investigation, using Computing to support learning in the same way that Mathematics supports the learning of subjects such as Physics." (p. 1) Whereas there has been a great body of research supporting these claims in the 1980s and 1990s (for an overview, please refer to Liao and Bright, 1991), it seems as if there is very little evidence on the transfer effects of computer programming skills in the context of twenty-first century education (Grover and Pea, 2013; Lye and Koh, 2014). Although computer programming and other skills

OPEN ACCESS

***Edited by:***
*Layne Kalbfleisch,*
*George Washington University, USA*

***Reviewed by:***
*Jonathan Plucker,*
*Johns Hopkins University, USA*

***Correspondence:**
*Ronny Scherer*
*ronny.scherer@cemo.uio.no*

***Specialty section:***
*This article was submitted to Educational Psychology, a section of the journal Frontiers in Psychology*

***Received:*** *31 July 2016*
***Accepted:*** *30 August 2016*
***Published:*** *14 September 2016*

***Citation:***
*Scherer R (2016) Learning from the Past–The Need for Empirical Evidence on the Transfer Effects of Computer Programming Skills. Front. Psychol. 7:1390. doi: 10.3389/fpsyg.2016.01390*

share a number of cognitive and even metacognitive processes (Clements, 1986, 1995; Brown and Kölling, 2012; Lye and Koh, 2014; Rich et al., 2014), therefore supporting potential transfer effects, I argue that educational research lags behind in sharing sufficient evidence for these claims.

Against this background, the main position this opinion paper conveys is that—although the conceptual argumentation about the potential transfer effects of computer programming skills on other skills such as problem solving and creativity is reasonable—there is a strong need for empirical evidence supporting this, particularly in the context of the recent advancements of digital technologies.

## CURRENT STATUS OF KNOWLEDGE

### Computer Programming Skills and the Concept of Computational Thinking

Computer programming skills are considered to be an integral part of what is called "computational thinking" (CT; Denning, 2010; Lye and Koh, 2014), and often find their way into frameworks of digital literacy (Siddiq et al., 2016). CT has in fact gained importance in STEM education, and there is a growing interest in exploring how CT can be introduced in K-12 curricula (Lye and Koh, 2014). Wing (2006) defined CT as a thought process that "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). In their review, Lye and Koh (2014) argued that computer programming—an activity that requires the abstraction and decomposition of problems—exposes students to these thinking processes, and claimed that it may therefore foster the development of CT. This claim has largely been supported by studies using the first (e.g., Logo®; Dyck and Mayer, 1989; Pardamean et al., 2015) and more recently developed programming tools for education (e.g., Scratch®; Wilson et al., 2013).

### Transfer Effects on Problem Solving

Having established that computer programming and CT are closely connected, the question arises to what extent transfer effects on problem solving exist. The starting point for addressing this question is to examine which particular thinking processes are involved in programming and problem solving. According to Brooks (1999), Clements and Merriman (1988), and Denning (2010), programming comprises a number of processes that range from information retrieval and processes of understanding the problem at hand to discovering methods and algorithms that solve the problem, and evaluating them (**Table 1**). According to a framework proposed by the (OECD, 2014), domain-general problem solving comprises similar processes of exploring and understanding, representing and formulating, planning, and executing, and monitoring and reflecting (**Table 1**). These conceptualizations reveal a considerable overlap between the two constructs: The processes involved in programming are by and large of the same nature as those involved in problem solving, in that the problem space needs to be understood and explored first, hypotheses and methods are to be developed second, and the proposed solution—be it an algorithm, product, or any other problem solution—finally needs to be evaluated (Klahr and Dunbar, 1988). This parallelism brings forth the question whether computer programming can actually be considered a form of problem solving (Barr and Stephenson, 2011; Jonassen, 2011), and suggests that transfer effects may exist. Investigating these ideas systematically, Liao and Bright (1991) conducted a meta-analysis of 65 studies that were conducted between 1969 and 1989 with the goal to synthesize empirical evidence on the effects of computer programming on problem solving abilities. The authors found an average effect size of 41 and concluded that students—while learning to program—acquire reasoning, logical thinking, and planning (i.e., problem solving) skills that go beyond computer programming. This meta-analysis was, however, followed by only a very limited number of experimental studies that continued examining these effects (Maloney et al., 2004; Gibbon, 2007; Pardamean et al., 2015), some of which were insignificant (Lai and Yang, 2011; Gülbahar and Kalelioğlo, 2014; Korkmaz, 2016). This is somehow surprising, because Liao and Bright (1991) clearly showed that the average effect was moderated by the type of programming environment used in the treatment group, thus suggesting that further advancements in these tools may affect the transfer effect. In fact, many publications that argued for the transfer effects later on only described programming tools or feasibility studies thereof on a conceptual level (Grover and Pea, 2013). By contrast, a larger body of research exists on the transfer effects on mathematical thinking and conceptual understanding (e.g., Calder, 2010; Kazakoff et al., 2013; Rich et al., 2014).

### Transfer Effects on Creativity

Similar to the reasoning on the transfer effects on problem solving, researchers have claimed that learning to program fosters students' creative thinking. In the most recent systematic review that I could identify (Clements, 1995), the author found that computer programming instruction fosters creativity and divergent thinking; significant effects on originality—a facet of creative thinking that involves selective encoding and combining—could be identified. Clements (1995) concluded his review by summarizing the key elements of computer programming that are also essential for creative thinking: decide on the nature of the problem, combine components of the problem, select a mental representation, monitor progress, acquire knowledge, and encode (see also Clements and Merriman, 1988). In a slightly different framework, Wallas (1926) proposed four stages of creativity, namely preparation, incubation, illumination, and verification, which are still widely used today. Comparing these processes with those required for computer programming (**Table 1**), substantive similarities exist, such that transfer effects between them could be expected. In fact, programming is considered to be a creative human activity (Denning, 2010; Grover and Pea, 2013). Yet, the existing body of empirical evidence supporting this expectation is rather meager: Except for a single study (Pardamean, 2014), the bulk of published empirical research on the transfer effects on creativity dates back to the 1980s and 1990s (Clements, 1986, 1991, 1995; McGrath, 1988; Subhi, 1999).

**TABLE 1 | Key processes involved in computer programming, problem solving, and creative thinking.**

| Computer programming | Problem solving | Creative thinking |
|---|---|---|
| ■ Representing, storing, and retrieving information in order to understand the problem (i.e., knowledge acquisition of basic problem elements such as objects, relations, initial and final states of objects)<br>■ Discovering algorithms for information processes —finding a method in order to represent the real-world problem, develop, and execute an action plan and code<br>■ Evaluating the performance of the designed complex systems on the basis of the written code; bridging the gap between the problem statement and the solution | ■ Exploring and understanding the problem (e.g., by decomposing the problem into sub-problems)<br>■ Representing and formulating the problem by creating representations of the problem situation and formulating hypotheses<br>■ Planning and executing the sequential steps to solve the problem<br>■ Monitoring progress and reflecting on the problem, the solution, and solution strategy | ■ Acquiring knowledge and skills relevant to the creative act, setting goals; encoding, recognizing, and formulating the problem (preparation)<br>■ Building a representation of the problem (e.g., by combining components of the problem) and unconscious processing (incubation)<br>■ Searching for and finding solutions (illumination)<br>■ Evaluating the creative product, monitoring the process of creative activities, and improving shortcomings (verification) |

## WHAT IS NEEDED

In light of this brief review of the current status of knowledge, I would like to follow Mayer (2015), who discussed the strong need for research evidence on game-based learning, and propose that research on the transfer effects of computer programming needs to move beyond untested claims and the mere description of programming tools, their feasibility, and students' interest or enjoyment thereof Fessakis et al. (2013) raised similar concerns: "Instead of focussing on the cognitive effects of programming, more recent studies concern the development of new programming environments for children [...]" (p. 89) Along these lines, educational research needs to focus more on the cognitive consequences of learning to program by designing experimental studies that systematically evaluate the transfer effects, making use of the well-developed programming tools such as Scratch® (Resnick et al., 2009). I believe that the outstanding number of feasibility studies on these tools create a fruitful ground for experimental comparisons. Such comparisons, however, need an appropriate research design, which fulfills at least three criteria (Mayer, 2015): (1) Appropriate *outcome measures* of academic learning or other skills that go beyond students' self-reports, enjoyment, or interest; (2) *Experimental control*—pretest-posttest designs with a treatment group (i.e., the group that learns to program) and a control group; (3) *Random assignment* of students to treatment and control groups. Despite the very few studies during the last 20 years, which adhered to these criteria (e.g., Pardamean et al., 2015), my observation is that research on the transfer effects of computer programming is in need of methodologically sound, experimental studies.

## CONCLUSION

From a conceptual perspective, the claims that learning computer programming may translate into the development of other, cognate skills such as problem solving or creative thinking, do have their standing, particularly because a considerable conceptual overlap between these skills exists. From an empirical perspective though, it seems as if the conceptual claims have been supplemented with evidence from experimental studies to a very limited extent. My observation is that most of the empirical research on the transfer effects dates back to the 1980s and 1990s; yet, too few studies have looked into these effects in the twenty-first century. This observation is somehow unexpected, particularly because Pea and Kurland (1984) pointed to the strong need for evidence on the transfer effects of computer programming that takes into account the development of digital technologies more than 20 years ago. Although this plea has been followed by a wave of experimental studies (Liao and Bright, 1991; Clements, 1995), educational research has not systematically followed up on examining the transfer effects. On the basis of the limited research on the transfer effects of computer programming skills on other cognitive skills on the one hand, and the conceptual claims that these transfer effects exist on the other hand, I would like to encourage researchers to fill this gap by learning from the past and reviving this research area in the twenty-first century.

## AUTHOR CONTRIBUTION

RS has conducted the literature review, drafted the manuscript, and performed revisions.

## REFERENCES

Barr, V., and Stephenson, C. (2011). Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? *ACM Inroads* 2, 48–54. doi: 10.1145/1929887.1929905

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., et al. (2012). "Defining Twenty-First Century Skills," in *Assessment and Teaching of 21st Century Skills*, eds P. Griffin, B. McGaw, and E. Care (Dordrecht: Springer), 17–66.

Brooks, R. (1999). Towards a theory of the cognitive processes in computer programming. *Int. J. Hum. Comput. Stud.* 51, 197–211. doi: 10.1006/ijhc.1977.0306

Brown, N., and Kölling, M. (2012). "Position paper: programming can deepen understanding across disciplines [DRAFT]," in *Paper Presented at the IFIP Working Conference–Addressing Educational Challenges: the Role of ICT* (Manchester, UK: Manchester Metropolitan University).

Calder, N. (2010). Using scratch: an integrated problem-solving approach to mathematical thinking. *Aust. Prim. Math. Classroom* 15, 9–14.

Care, E., and Anderson, K. (2016). *How Education Systems Approach Breadth of Skills*. Washington, DC: Center for Universal Education at BROOKINGS.

Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *J. Educ. Psychol.* 78, 309–318. doi: 10.1037/0022-0663.78.4.309

Clements, D. H. (1991). Enhancement of creativity in computer environments. *Am. Educ. Res. J.* 28, 173–187. doi: 10.2307/1162883

Clements, D. H. (1995). Teaching creativity with computers. *Educ. Psychol. Rev.* 7, 141–161. doi: 10.1007/bf02212491

Clements, D. H., and Merriman, S. (1988). "Componential developments in LOGO programming environments," in *Teaching and Learning Computer Programming: Multiple Research Perspectives*, ed R. E. Mayer (Hillsdale: Lawrence Erlbaum Associates, Inc.), 13–54.

Denning, P. J. (2010). Great principles of computing. *Am. Sci.* 98, 369–372. doi: 10.1511/2010.86.369

Dyck, J. L., and Mayer, R. E. (1989). Teaching for transfer of computer program comprehension skill. *J. Educ. Psychol.* 81, 16–24. doi: 10.1037/0022-0663.81.1.16

Fessakis, G., Gouli, E., and Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: a case study. *Comput. Educ.* 63, 87–97. doi: 10.1016/j.compedu.2012.11.016

Gibbon, L. W. (2007). *Effects of LEGO Mindstorms on Convergent and Divergent Problem-Solving and Spatial Abilities in Fifth and Sixth Grade Students (Doctor of Education)*. Seattle, WA; Seattle Pacific University.

Greiff, S., Wüstenberg, S., Csapó, B., Demetriou, A., Hautamäki, J., Graesser, A. C., et al. (2014). Domain-general problem solving skills and education in the 21st century. *Educ. Res. Rev.* 13, 74–83. doi: 10.1016/j.edurev.2014.10.002

Grover, S., and Pea, R. (2013). Computational thinking in K-12: a review of the state of the field. *Educ. Res.* 42, 38–43. doi: 10.3102/0013189x12463051

Gülbahar, Y., and Kalelioğlo, F. (2014). The effects of teaching programming via Scratch on problem solving skills: a discussion from learners' perspective. *Inform. Educ.* 13, 33–50.

Jonassen, D. H. (2011). *Learning to Solve Problems: A Handbook for Designing Problem-Solving Learning Environments*. New York, NY: Routledge.

Kay, K., and Greenhill, V. (2011). "Twenty-First century students need 21st century skills," in *Bringing Schools into the 21st Century*, eds G. Wan and M. D. Gut (Dordrecht: Springer), 41–65.

Kazakoff, E. R., Sullivan, A., and Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Child. Educ. J.* 41, 245–255. doi: 10.1007/s10643-012-0554-5

Klahr, D., and Dunbar, K. (1988). Dual space search during scientific reasoning. *Cogn. Sci.* 12, 1–48. doi: 10.1207/s15516709cog1201_1

Korkmaz, Ö. (2016). The effect of scratch- and lego mindstorms Ev3-Based programming activities on academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *Malays. Online J. Educ. Sci.* 4, 73–88.

Lai, A.-F., and Yang, S.-M. (2011). "The learning effect of visualized programming learning on 6 th graders' problem solving and logical reasoning abilities," in *Paper presented at the International Conference on Electrical and Control Engineering (ICECE)* (Yichang).

Liao, Y.-K. C., and Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: a meta-analysis. *J. Educ. Compu. Res.* 7, 251–268. doi: 10.2190/e53g-hh8k-ajrr-k69m

Lye, S. Y., and Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: what is next for K-12? *Comput. Human Behav.* 41, 51–61. doi: 10.1016/j.chb.2014.09.012

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004). "Scratch: A sneak preview," in *Paper presented at the Second International Conference on Creating, Connecting and Collaborating Through Computing* (Kyoto).

Mayer, R. E. (2015). On the need for research evidence to guide the design of computer games for learning. *Educ. Psychol.* 50, 349–353. doi: 10.1080/00461520.2015.1133307

McGrath, D. (1988). Programming and problem solving: will two languages do it? *J. Educ. Comput. Res.* 4, 467–484.

OECD (2014). *PISA 2012 Results: Creative Problem Solving: Students' Skills in Tackling Real-Life Problems*, Vol. 5. Paris: OECD Publishing.

Pardamean, B. (2014). Enhancement of creativity through logo programming. *Am. J. Appl. Sci.* 11, 528–533. doi: 10.3844/ajassp.2014.528.533

Pardamean, B., Suparyanto, T., and Evelyn. (2015). Improving problem-solving skills through Logo programming language. *New Educ. Rev.* 41, 52–64. doi: 10.15804/tner.2015.41.3.04

Pea, R. D., and Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas Psychol.* 2, 137–168.

Pellegrino, J. W., and Hilton, M. (2012). *Education for Life and Work: Developing Transferable Knowledge and Skills in the 21st Century*. Washington, DC: The National Academies Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Commun. ACM* 52, 60–67.

Rich, P. J., Bly, N., and Leatham, K. R. (2014). Beyond cognitive increase: investigating the influence of computer programming on perception and application of mathematical skills. *J. Comput. Math. Sci. Teach.* 33, 103–128.

Scherer, R. (2015). Is it time for a new measurement approach? A closer look at the assessment of cognitive adaptability in complex problem solving. *Front. Psychol.* 6:1664. doi: 10.3389/fpsyg.2015.01664

Siddiq, F., Hatlevik, O. E., Olsen, R. V., Throndsen, I., and Scherer, R. (2016). Taking a future perspective by learning from the past – A systematic review of assessment instruments that aim to measure primary and secondary school students' ICT literacy. *Educ. Res. Rev.* 19, 58–84. doi: 10.1016/j.edurev.2016.05.002

Sturman, L., and Sizmur, J. (2011). *International Comparison of Computing in Schools*. Slough: National Foundation for Educational Research.

Subhi, T. (1999). The impact of LOGO on gifted children's achievement and creativity. *J. Comput. Assist. Learn.* 15, 98–108. doi: 10.1046/j.1365-2729.1999.152082.x

Tricot, A., and Sweller, J. (2013). Domain-specific knowledge and why teaching generic skills does not work. *Educ. Psychol. Rev.* 26, 265–283. doi: 10.1007/s10648-013-9243-1

Wagner, T. (2012). *Creating Innovators - The Making of Young People Who Will Change the World*. New York, NY: Scribner.

Wallas, G. (1926). *The Art of Thought*. New York, NY: Harcourt, Brace and Company.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., et al. (2016). Computer science in K-12 school curricula of the 2lst century: why, what and when? *Educ. Inf. Technol.* doi: 10.1007/s10639-016-9493-x. [Epub ahead of print].

Wilson, A., Hainey, T., and Connolly, T. M. (2013). Using Scratch with primary school children: an evaluation of games constructed to gauge understanding of programming concepts. *Int. J. Game-Based Learn.* 3, 93–109. doi: 10.4018/ijgbl.2013010107

Wing, J. M. (2006). Computational thinking. *Commun. ACM* 49, 33–35. doi: 10.1145/1118178.1118215

**Conflict of Interest Statement:** The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.