Author's Response

De-Constructionism: Practice, Examples, Bugs

Jean M. Griffin
Temple University, USA
jeaniacgriffin/at/gmail.com

> Abstract • This response clarifies what de-constructionism is and explains how it differs from, complements, and to some extent overlaps with constructionism. It clarifies the distinction between the terms de-construction and deconstruction, and discusses student motivation and social interaction.

« 1 » Seymour Papert's protégé Idit Harel describes constructionism as follows:

> "Seymour coined the term to advance a new theory of learning, claiming that children learn best when they
> 1. use tech-empowered learning tools and computational environments,
> 2. take active roles of designers and builders; and
> 3. do it in a social setting, with helpful mentors and coaches, or over networks."[1]

« 2 » De-constructionism complements constructionism and is applicable to students of all ages. It has three guiding principles: provide ample, effective practice; guide students to deconstruct well-built examples in ways that promote metacognition; and develop negative knowledge by carefully crafting intentional bugs.

« 3 » Motivation is an important dimension of learning. To answer **Mihaela Sabin**'s Q2, the model for de-construction focuses on cognition rather than motivation. One way to address motivation is to pair de-constructionist activities with constructionist ones. Another way is to give students interesting things to deconstruct – well-built things that students are excited to learn about. These can be big things or physical things; they need not be limited to code segments as used in this study. Yet another way to foster motivation is to gamify the experience with rewards such as points, levels, or badges.

« 4 » Social interaction is another important dimension of learning that is critical to constructionism. Again, one way to address social interaction (Sabin Q3) is to pair de-constructionist activities with constructionist ones. This can also be accomplished through competitions or with a variety of collaborative approaches used in CS education. For example, with Pair Programming, students take turns using the computer keyboard. With Peer Instruction, the teacher poses a multiple-choice question. Students vote for an answer, discuss the question with their classmates, and vote again. Then the teacher reveals the correct answer. When designing Pair Programming or Peer Instruction activities, instructional designers can use the model for de-construction as a guide. POGIL (Process Oriented Guided Inquiry Learning) uses structured small-group learning where each student is assigned a role (e.g., manager, recorder, reporter). The group is guided to explore (deconstruct) a model via a series of questions, and then apply what they have learned by completing challenges. A contribution that de-construction makes to supplement POGIL is its emphasis on practice.

« 5 » I agree with the hypothesis that the de-constructionist approach can promote learning (**Sabin** Q1). Typically, learning is measured with pre/post assessments, where learning gains are calculated as the difference between pre-test and post-test scores (individually for each student or in aggregate for a collection of students). The challenge with evaluating de-construction is that it has so many components. Not only does it promote a variety of techniques (e.g., explaining, labeling, comparing, completing), it also promotes variation in practice (concrete-to-abstract, distributed, interleaved). This makes it challenging to design a controlled study. If there are differences in learning gains between a de-construction group and a control group, to which of these features can the differences be attributed? There is also the longitudinal factor – designs learned through de-construction may only prove useful much later during a design process. Because of these complexities I conducted an experiment to evaluate just one aspect of de-construction – learning from intentional bugs. Within the study described in this target article, I conducted an experiment where half the students got some practice problems with bugs while the other half got similar problems without bugs. Learning gains were measured with a pre/post placement test and exams. This experiment is discussed in a forthcoming article (Griffin 2019) based on my dissertation (Griffin 2018), accompanied by a collection of design principles to supplement the model for de-construction presented in this target article.

« 6 » To answer **Pavel Boytchev**'s Q2 about whether there have been situations in which de-constructionism has had a negative impact, I will share a personal experience. Several years ago, my team developed practice problems called *Debug'ems*™ that challenged students to find and fix bugs. When we gave them to advanced high-school students in a five-week summer program, the students enjoyed the challenge, especially because they involved animations with dramatic themes and the bugs had humorous consequences. When we gave the same problems to another group of high-school students – economically disadvantaged ones participating in short workshops – these students found the problems to be too difficult. Some would say that this was due to cognitive overload. I became aware that in this context, asking students to find and fix bugs was too difficult. It had the negative consequence of discouraging the students unnecessarily. To address this concern, I designed alternative activities similar to treasure hunts that provided more scaffolding. I created fun animations with code that the students were not yet capable of writing on their own. Students were guided to explore the code and answer challenging questions about it. These *Explore'ems* provided more scaffolding than the Debug'ems™ and the students enjoyed them (Griffin, Kaplan & Burke 2012). In the model for de-construction, Explore'ems would fall on the easier, left-hand side of the model, with the understanding that it is certainly possible to create some Debug'ems™ that are easier to solve than some Explore'ems.

« 7 » It is worth discussing the differences between de-construction and deconstruction. As to whether the impulse to deconstruct – take things apart – is an in-

1| "A glimpse into the playful world of Seymour Papert" by Idit Harel, EdSurge, 2016. https://www.edsurge.com/news/2016-08-03-a-glimpse-into-the-playful-world-of-seymour-papert

born human behavior like curiosity or creativity (Boytchev Q1), I am inclined to think so, but I will leave that for developmental psychologists to debate. I appreciate that James Clayson (§§16–18) mentions Jacques Derrida. Although I refer to Derrida in my dissertation (Griffin 2018) I did not do so in my target article. Derrida's philosophy of literary deconstruction is well established. This makes it problematic to use the term deconstructionism in another sense, as a pedagogy complementary to constructionism, as I initially did in 2012 (Griffin, Kaplan & Burke 2012: 4). I now use the term de-construction to clarify the distinction.

« 8 » Although de-constructionism's learning-by-taking-apart approach stands in contrast to constructionism, it makes explicit some of Papert's ideas and values that are frequently overlooked. For example, Papert valued learning by taking things apart. He recounted transformative childhood experiences of taking apart gears (Papert 1980). De-constructionism explicitly highlights learning-by-taking-apart by promoting specific techniques informed by cognitive psychology. These include explaining, labeling, comparing, discerning, and completing. Papert also viewed debugging as a powerful idea (ibid). De-constructionism explicitly promotes learning from bugs and the acquisition of negative knowledge, not just by learning from one's own mistakes, but by interacting with carefully designed intentional bugs. It also emphasizes effective practice, which is not addressed by constructionism. Practice with taking things apart and negative knowledge are critical in learning environments intended to develop technical expertise. Thus, although de-constructionism overlaps with constructionism to some extent, it is not redundant (Clayson §2), and there is not a strict dichotomy between the two paradigms (Clayson §1). Although constructionism was designed for young children, some argue that bringing it to older students is warranted (Laurillard 2002; Sacristán et al. 2018). Combining constructionism with de-constructionism is an effective way to bring constructionism to older students.

« 9 » Clayson's remarks (§10) about instructionism indicate a misunderstanding. I am a big fan of constructionism, especially in science and math courses. In settings where it is important for students to develop technical expertise, however, pairing it with de-construction will achieve the goals of topic coverage and skill development better than constructionism alone. To clarify, characterizing de-constructionism as opposite to constructionism (e.g., in the title of the target article) refers to the opposing approaches of learning-by-making and learning-by-taking-apart. This is a commonsense distinction that is accessible to teachers. De-constructionism is not meant to champion what Papert disdained about the ways in which math is typically taught in schools, devoid of personal utility, or *thingness*, or interest – all aspects of instructionism (Papert 1993, 1996). De-constructionism is inspired by hands-on activities with concrete things – by reverse engineering, mechanical dissection, and Tod Phod Jod. The main idea is to give students exposure to well-built examples that are relevant to them, which they can assimilate as design patterns and use later for their own constructions.

Acknowledgments

I thank the reviewers and editors for their thoughtful comments as well as my dissertation committee members, Catherine Schifter, Kristie Newton, John Dougherty, and Julie Booth.

References

Griffin J. M. (2018) Learning to program with interactive example code (with and without intentional bugs). Doctoral dissertation, Temple University, USA.

Griffin J. M. (2019) Designing intentional bugs for learning. In: Proceedings of the first UK and Ireland Computing Education Research conference. ACM, Canterbury UK, in press.

Griffin J. M., Kaplan E. & Burke Q. (2012) Debug'ems and other Deconstruction Kits for STEM learning. In: Proceedings of the second IEEE integrated STEM education conference (ISEC '12). IEEE Press, Piscataway NJ: 1–4.

Laurillard D. (2002) Rethinking university teaching: A conversational framework for the effective use of learning technologies. Second edition. Routledge, London.

Papert S. (1980) Mindstorms: Children, computers, and powerful ideas. Basic Books, New York.

Papert S. (1993) The children's machine: Rethinking school in the age of the computer. Basic Books, New York.

Papert S. (1996) An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning 1(1): 95–123.

Sacristán A. I., Kaminskiene L., Sabin M. & Baafi R. A. K. (2018) Constructionism in upper secondary and tertiary levels. In: Dagienė V. & Jasutė E. (eds.) Constructionism 2018. Vilnius University, Vilnius: 932–946.

Received: 30 June 2019
Accepted: 10 July 2019