Open Peer Commentaries

on Pavel Boytchev's "Constructionism and Deconstructionism"

Deconstruction in Software Construction

Gerald Futschek
Vienna University of Technology, Austria
gerald.futschek/at/tuwien.ac.at

> Upshot • Boytchev's deconstructionism looks at first glance like a game of words. Upon a deeper view of the subject, he focuses our attention on the importance of deconstruction to the construction process, which is highly connected to creativity. In my contribution, I want to point out the close relationship of Boytchev's deconstruction to the software development process, where requirements analysis corresponds to deconstruction and software design and implementation correspond to construction. Creativity is an important asset in any kind of software development, where life-long learning is essential.

« 1 » At first glance, the article's title was provocative to me. I felt that deconstruction, as the opposite of construction, means something like destruction. How can destruction help to construct knowledge? I was aware that destruction is the contrary of construction and that destruction is sometimes a necessary prerequisite of construction. Shiva, one of the major deities of Hinduism, is known not only as destroyer but also as creator. Often the need for a construction arises from a destruction. So a close relationship between construction and destruction exists.

« 2 » Pavel Boytchev gives a wise definition of deconstruction: a decomposition of something into reusable entities (§24). These reusable entities can be used in the following construction phase. So deconstruction is not the opposite of construction, it is an essential part of it and incorporates a great deal of necessary creativity. Boytchev's definition of deconstruction is therefore completely different to what can be expected from the title.

« 3 » And why create a new –ism out of deconstruction (§42)? Here, I think it is not justified to define deconstructionism and use it in the same line as constructionism. It is, from my point of view, really a game of words. Deconstruction is, by definition, part of the construction process and therefore subordinate to construction.

« 4 » Boytchev's description of the phases of learning through deconstruction and construction strongly reminds me of core activities of software development processes: requirements analysis, software design and software construction. For more details on software development processes, see, for example, the recent book by Murali Chemuturi (2012). When a larger piece of software has to be developed, first of all the problem domain has to be understood by the developers. The result of analysing the requirements is a structured model of the problem domain. This is usually a decomposition of the problem domain into smaller entities that interact with each other. The main purpose of breaking it up into its component parts is not only the better understanding of the problem domain but also the reuse of the parts in the constructive activities of software design and software construction. In the software design activity, a plan for the construction of the software is created. A part of this plan results from the components of the requirement analysis.

« 5 » Creativity is an essential property of software development. For each problem, there are endless potential software solutions that tackle the problem. It is a creative act to find a better solution that is highly-appreciated by users. Not all steps of the software development process can be performed just by following well-defined rules. The experience and creativity of the software engineer strongly influences the outcome of his activities. The way we look at the problem domain and how we deconstruct it into manageable parts strongly influences the quality of the final software product.

« 6 » Why do I connect constructionist learning with software engineering? It is not only the similarity of the processes. It is also the fact that there is constructionist learning in the software industry. It is a life-long learning process that forms good software engineers. They learn from their own and other's drawbacks and successes. So the software engineering activities can be seen as a life-long constructionist learning process.

« 7 » It is not only the future of learning that is massively influenced by the exponential growth in technology (§46), constructionism may also positively influence information technology with its potential in the life-long learning process of software engineers.

« 8 » The two practical examples of constructionist learning that are presented at the beginning of Boytchev's article show very impressively how technology may stimulate the learner's creativity. These technologies are very powerful; they allow a large variety of learning paths and so make room for creativity. The library Mecho (§6)is a very flexible programmable tool that allows

university students to create wonderful new virtual mechanisms. Completely different is the tool that was designed for school students to explore a virtual classroom (§17). The students can also provoke very extreme weather situations and study their effect on the virtual classroom. Furthermore, this simulation tool allows a variety of inquiry-based learning experiences that support the creative thinking of the learners.

Gerald Futschek is professor at the Institute of Software Technology and Interactive Systems at the Vienna University of Technology and chair of the special interest group on teacher education of the Austrian Computer Society. His areas of interest include software engineering, program verification and informatics didactics. Homepage: http://www.ifs.tuwien.ac.at/futschek

Received: 15 June 2015
Accepted: xxxx

Construction and Deconstruction

Brian Harvey
University of California, Berkeley, USA • bh/at/cs.berkeley.edu

> Upshot • Pavel Boytchev's article calls attention to the fruitful dialectic between building things and taking them apart: No successful construction without deconstruction. Of course by using the word "deconstruction," he is also implicitly invoking the critical-theory sense of the term, inviting us to deconstruct constructionism. I found the article fascinating on both levels.

« 1 » This is a very provocative target article. Its central thesis, that deconstruction is essential to successful construction, is an idea that, once stated, is obvious, but it is far out of the mainstream of constructionist vocabulary. I think, though, that the *practice* of deconstruction may be more prevalent than its analysis.

« 2 » For one thing, it is a commonplace observation that you learn how to build things by looking at, and looking inside, already-built examples. Sometimes this involves physical deconstruction, as in Pavel Boytchev's example of the child deliberately breaking a toy to see how it works. But I am thinking also of the "Look inside" button on Scratch project pages. The visitor's first view of someone else's Scratch project focuses on its behavior and purpose. But by clicking "Look inside," the visitor can examine the Scratch program that makes the project function, and can even modify or "remix" it.

« 3 » Secondly, people try to build learning environments based on a small number of simple parts. The canonical example is Lego, which has been much more successful than the Erector sets of my own childhood – in which the main components were metal bars of many different lengths that could be bolted together, along with specialized connectors such as angle brackets – in part because of the wide range of projects that can be built out of (many copies of) a single part, the Lego brick. Such environments come pre-deconstructed. The same could be said of the effort in the design of Scratch to minimize block shapes and to minimize the number of primitive blocks altogether.

« 4 » Thirdly, the emphasis on physical stuff in the Maker Movement makes the component parts much more readily visible than in either computer software or even more abstract mathematics.

« 5 » I would be interested in a discussion of how the idea of deconstruction fits in with the (I think) overlapping idea of demystification. One way to demystify a black box is to see how it is built. But there are also demystification strategies that, for example, try to shed light on people's motivation, or on sociological ideas such as "institution." Is the purpose of deconstruction demystification, or something else?

« 6 » I think some of the points in the article are made in too much of a hurry. Here are two examples.

« 7 » Debugging as deconstruction (§32): I understand the overall point, I think; in order to debug you have to think analytically about the pieces of a program. And I get that current debugging technology is weak. But I do not think I quite got how a deconstructionist perspective can lead to better debugging tools. This could be a research project in itself, and the one paragraph here leaves me dissatisfied.

« 8 » Technology and education (§45): There is a claim here that very rapid change in technology makes it hard for education to keep up. I see how that would apply for education about technology – which is sort of the point of the Noam Chomsky quotation in §45. And I can see how education might not be making the best possible use of technology: although, for example, schools have been quick to jump on the tablet bandwagon, as far as I know they have not yet found any purpose for which tablets are better than laptops. (I do get that they are cheaper, although now you can get netbooks instead.) But children are not so very different in how they construct knowledge, are they? If that is the intended claim, it needs much more discussion. Otherwise the point about exponential change is a little glib.

« 9 » In contrast, the discussion of the virtual classroom project is detailed, vivid, and very convincing. Figures 9 and 10 are a wonderful encapsulation of the central point. I would have been glad if the last few pages of the article were more a laying out of a specific program for future research, rather than a collection of brief mentions of largely unsupported ideas.

« 10 » I do wonder why students of the virtual classroom were required to deconstruct not only the physical model being simulated but also the simulator as a piece of software. Why, for example, make a point of not documenting which keystroke does what in the user interface? Does *that* deconstruction task not just distract from the deconstruction of the classroom environment?

« 11 » There is another paper to be written about the deconstruction of education. Does that mean something more than the critique of education that starts (maybe) with Jean Jacques Rousseau and goes through John Dewey to Paulo Freire and Ivan Illich? Is there some way in which computers change this critique? What *are* the component parts of education? Learner, teacher, and content? Maybe plus the learning community? Or does Boytchev mean something more detailed than that? When I visit a school, I generally find myself thinking first not about technology, but