Constructivism in Computer Science Education

Mordechai Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
ntbenari@wis.weizmann.ac.il

Abstract

Constructivism is a theory of learning which claims that students construct knowledge rather than merely receive and store knowledge transmitted by the teacher. Constructivism has been extremely influential in science and mathematics education, but not in computer science education (CSE). This paper surveys constructivism in the context of CSE, and shows how the theory can supply a theoretical basis for debating issues and evaluating proposals.

Introduction

The dominant theory of learning today is called *constructivism*. This theory claims that knowledge is actively constructed by the student, not passively absorbed from textbooks and lectures. Since the construction builds recursively on knowledge (facts, ideas and beliefs) that the student already has, each student will construct an idiosyncratic version of knowledge. Teaching techniques derived from the theory of constructivism are supposed to be more successful than traditional techniques, because they explicitly address the inevitable process of knowledge construction.

Constructivism has been intensively studied for more than a decade by researchers of science education [7] and mathematics education [1]. This article will first survey constructivism and its application in science education; then I will assess its relevance for CSE and offer some examples of the application of the theory.

Previous Work

There is a large literature on the psychology of programming ([17], [10]), and many of these researchers clearly hold constructivist views. For example, the Boxer system was specifically designed on constructivist principles ([2], p. 12). However, I could not find articles on constructivism in computer science *education* comparable with the vast literature in mathematics and physics education.

SIGSCE 98 Atlanta GA USA

Epistemology and Constructivism

An educational paradigm is composed of four components [5]:

- An *ontology* which is a theory of existence.
- An *epistemology* which is a theory of knowledge, both of knowledge specific to an individual and of shared human knowledge.
- A *methodology* for acquiring and validating knowledge.
- A *pedagogy* which is a theory of teaching.

We can use this framework to describe the classical educational paradigm:

- There *is* an ontological reality. The Newtonian model of absolute space and time is the model of reality we use in practice, and we are Platonists who hold that mathematics has an independent existence.
- Epistemology is *foundational*. The truth is out there. Through empirical experiences we can discover absolutely true foundations, and use valid forms of logical deduction to expand true knowledge.
- The mind is a clean slate that can be filled with transmitted knowledge.
- Listening to lectures and reading books are the primary means of knowledge transmission. Repetition will ensure that the knowledge is retained.

The constructivist paradigm is dramatically different:

- Ontological reality is at best irrelevant. Since we can never truly 'know' anything, ontology cannot influence our educational paradigm.
- The epistemology of constructivism is *nonfoundationalist* and *fallible*. Absolute truth is unattainable,

so there is no foundation of truth on which to build. Knowledge is constructed by each individual and thus necessarily fallible.

- Knowledge is acquired recursively: sensory data is combined with existing knowledge to create new *cognitive structures*, which are in turn the basis for further construction. Knowledge is also created cognitively by *reflecting* on existing knowledge.[1]
- Passive learning will likely fail, because each student brings a different knowledge framework to the classroom, and will construct new knowledge in a different manner. Learning must be active: the student must construct knowledge assisted by guidance from the teacher and feedback from other students.

The task of the teacher is significantly more difficult than in the classical paradigm, because the guidance must be based on the understanding of each student's currently existing cognitive structures.

Constructivists differ among themselves as to the relative importance ascribed to the individual student and to the group in constructing knowledge; these variants are known as 'radical' and 'social' constructivism, respectively. A discussion of this topic is beyond the scope of this article (see [5]).

Constructivism in Science Education

Studies have shown that relatively few students reach an acceptable level of achievement in science and mathematics [4]. Physics teachers seem to have the worst time, as students retain a naive theory of physics despite intensive instruction in Newtonian mechanics. Apparently, these ideas are so entrenched that mere lectures and even experiments have a difficult time evicting them.

According to constructivism, a teacher cannot ignore the student's existing knowledge; instead, he or she must question the student in order to understand what models the student possesses, and only then attempt to guide the student to the 'correct' theory. A term that is commonly used is *alternative frameworks*, meaning that the students have consistent models—they just happen to be at variance with the (currently accepted) scientific theory. Similarly, von Glasersfeld would never say that a student's knowledge is 'wrong', because he does not believe in the possibility of establishing truth. Instead, he says that concepts are *viable* 'if they prove adequate in the contexts in which they were created' ([20], p. 7).

In most fields of science education (including CSE), there is a large body of research that catalogs *misconceptions*. A constructivist would view a misconception not as a mistake, but as a logical construction based on consistent, though non-standard concepts, held by the student. Misconceptions form the prior knowledge that is essential to the construction of new knowledge [16]. Merely listing misconceptions is fruitless; a description of the underlying model and a prescription for constructing a modified one must be given.

Criticism of Constructivism

Before continuing, we must stress that that there is opposition to constructivism.[2] The criticism is not so much of the constructivist theory of learning, but rather of extreme conclusions drawn from constructivist epistemology. Carried to the extreme, radical constructivism leads to *solipsism*, the philosophical claim that the world is my own creation or dream. In turn, this can lead to a rejection of ethics: if the world is my own creation, why should I care what happens to others? Carried to the extreme, social constructivism leads to a view of science as a merely political enterprise developed by entrenched elitist groups whose main purpose is to ensure their own survival. From the fallibility of scientific knowledge, one slips into relativism of truth, and from the sociology of scientific practice, into demands for empowerment detached from any attempt at objective evaluation of scientific knowledge.

Empirical Results in CSE

There is no question that many students find the study of computer science extremely difficult, especially at elementary levels. Before proceeding with a theoretical analysis, I wish to quote some empirical results that demonstrate the depth of the problem:

- Sleeman *et al.* [15] found that the concept of *variable* is extremely difficult for students. For example, students taught a 'box'-metaphor believe that a variable can simultaneously contain two values. The students have *constructed a consistent concept*, it just happens to be non-viable for successful programming.
- Madison [9] used extensive interviews to elicit the internal—non-viable—models of parameters held by students.
- Holland *et al.* [8] show the extent of the misconceptions held by students studying object-oriented programming.

Constructivism in CSE

To what extent is constructivism applicable to CSE? According to constructivism, students construct knowledge by combining the experiential world with existing

[1] These terms come from the seminal work of Jean Piaget on the acquisition of knowledge by children; Piaget's work was instrumental in the development of constructivist theories.

[2] For more detail see the articles by Matthews, Nola, Phillips and Ogborn in the Special Issue on Philosophy and Constructivism in Science Education of *Science & Education*, January 1997.

cognitive structures. I claim that the application of constructivism to CSE must take into account two characteristics that do not appear in natural sciences:

- A (beginning) CS student has no *effective model* of a computer.
- The computer forms an *accessible ontological reality*.

By effective model, I mean a cognitive structure that the student can use to make viable constructions of knowledge, based upon sensory experiences such as reading, listening to lectures and working with a computer. By accessible ontological reality, I mean that a 'correct' answer is easily accessible, and moreover, successful performance requires that a normative model of this reality must be constructed.

The following paragraphs expand on these claims.

The naive theory of physics held by students is clearly effective, but I deny that beginning CS students come to class with an effective model of a computer. At most, the model is limited to the grossly anthropomorphic 'giant brain', hardly a useful metaphor when studying computer science. Pea [11] gives the name 'superbug' to the idea that a 'hidden mind' within the programming language has intelligence. If misconceptions are essential to the construction of new knowledge [16], the lack of an effective, if flawed, model of a computer is a serious obstacle to learning CS.

At the novice level, the claim is supported by many studies. Du Boulay ([3], p. 285) notes that "Even if no effort is made to present a view of what is going on 'inside' the learners will form their own." Perkins *et al.* ([12], p. 162) "attribute students' fragile knowledge of programming in considerable part to a lack of a mental model of the computer...." Sleeman *et al.* ([14], p. 251) found that "even after a full semester of Pascal, students' knowledge of the conceptual machine underlying Pascal can be very fuzzy."

The computer science student is faced with immediate and brutal feedback on conclusions drawn from his or her internal model. More graphically, *alternative frameworks cause bugs*. Computer science is unlike school physics: intuition and manipulative facility are not sufficient for passing a course, and the consequences of misconceptions are immediately exposed. There is not much point in arguing about 'alternative frameworks' of the syntax or semantics of a programming language. Surely, this is the cause of much psychological grief experienced by computer science students.

Application of Constructivism in CS

I believe that many phenomena of CSE can be explained by constructivism:

- The construction of CS concepts is haphazard (as shown by the research cited), because sensory data from class must be integrated into a student's existing framework that is too superficial.
- Frustration and the perception that computer science is 'hard' is due to the fact that models must be self-constructed from the ground up.
- Autodidactic programming experience is not necessarily correlated with success in academic CS studies. These students, like physics students, probably come with firmly held constructions that are not viable for academic studies.
- The reality feedback obtained by working on a computer can be discouraging to students who prefer a more reflective or social style of learning.

In the rest of the paper, I will apply constructivist principles to specific issues in CSE. The point is not necessarily to convince you that my analyses and conclusions are correct, but to convince you that constructivism can provide a new and powerful set of concepts to guide our debates on CSE.

GUI and WYSIWYG Angst

Graphical user interfaces (GUI) are often touted as 'intuitive' and 'user-friendly', yet many new users are anxiety-ridden. From a constructivist point of view, this paradox is easily resolved. An icon is just a representation; it is useful only to the extent that the user can *construct* a mental model of object being represented.

Consider the icon for a 'running man'. This represents the action that 'causes the computer to perform the machine instructions that are obtained by compiling the source code'. Jargon calls this 'running a program'. The icon is thus two steps removed from the action: first, the icon must be deciphered as representing the word 'running' rather than 'chasing' or 'racing', and then this word must be related to a mental model of the computer.

Icons are intuitive to the extent that the analogy between the object shown and the object represented is perfect. But as Glynn shows [6], analogies are rarely, if ever, perfect, so one must not lose patience with a novice who has yet to construct an efficient model of the underlying machine.

WYSIWYG (What You See Is What You Get) is another concept that could benefit from some constructivist analysis. Opponents of WYSIWYG often paraphrase it sarcastically as 'what you see is all you get', but this misses the point. What you see is *not* what you get: what you *get* is an internal data structure containing your document and a set of operations

for transforming the data structure; what you *see* is merely a visual representation of the structure, and icons or menus representing the operations. You have to *construct* a viable model that will enable you to *predict* the outcome of any operation on the model, and to predict how that outcome will be reflected in the representation you see. The relevance for CSE is that courses, help files and tutorials must *explicitly* address the construction of a model, and not limit themselves to behaviorist practices of the form 'to do X, following these steps'.

Explicitly Teach the Model

If the student does not bring a preconceived model to class, then we must ensure that a viable hierarchy of models is constructed and refined as learning progresses. This means that the model of a computer—CPU, memory, I/O peripherals—must be *explicitly* taught and discussed, not left to haphazard construction and not glossed over with facile analogies. Teaching the model can be done using *epistemic games*—formalized procedures for constructing knowledge—such as a model computer [13] or a 'notional machine' [3]. From a constructivist perspective, the appropriate level of an initial model is a matter for empirical research: given that a model of a computer must be taught, what is the most abstract model that will be effective.

My conclusion that a model of the computer be explicitly taught may not be compatible with teaching objects in introductory courses. Abstraction is essential as a way of 'forgetting' detail, and software development would be impossible without it, but it seems to me that there must be an *object-oriented paradox*: how is it possible to forget detail that you never knew or even imagined?

I appreciate the attractiveness of an objects-first approach; the gap between the standard libraries (especially the GUI libraries) of a modern programming environment, and the model of a computer is so great that motivating beginners has become a serious problem. My question is: if students find it difficult to construct a viable model of variables and parameters, why should we believe that they can construct a viable model of an object such as a radio button? Teachers of introductory courses that use abstractions must not assume that students will construct the model that the instructor has. This was demonstrated by Holland *et al.* [8], but based on experience in other disciplines of science education, cataloging and analyzing misconceptions will not be sufficient to improve students' understanding.[3]

[3]Turkle and Papert claim that OOP is "not only more congenial to those who favor concrete approaches, but it also puts an intellectual value on a way of thinking that is resonant with their own." ([19] p. 155) This claim is strange, because the point of studying OOP is to learn to create abstractions, not just to use existing objects.

Bricolage and Gender

Bricolage is a term coined by the anthropologist Claude Lévi-Strauss, who used it in a derogatory sense for the 'science of the concrete' in primitive societies, as opposed to abstract 'European' science. Turkle and Papert [19] transferred the concept to the context of learning to program, and vehemently defend it as a learning style as valid as the normative 'planning' style that we attempt to teach.

This is consistent with a constructivist view of education: different students will approach the construction of knowledge in different ways, and the educational environment must be supportive of these differences. Certainly, concrete thinking and bricolage can help students in introductory courses. However, I do not think that bricolage is an effective epistemology for professional programming. A massive amount of detailed knowledge must be constructed, so it must be organized in levels of abstraction (cf. object-oriented programming), and the normative 'planning' style that we call software engineering must eventually be learned and practiced.

A manifestation of bricolage in computer science is endless debugging of the 'try-it-and-see-what-happens' variety. A student who *exclusively* uses such techniques is ultimately not qualified to work on the software of embedded and operating systems, which requires the ability to create and test abstract hypotheses.

Turkle and Papert published their article arguing for tolerance of concrete thinking in a journal subtitled *Women in Culture and Society*, and they chose two women to exemplify college students who are concrete thinkers. I believe that the concrete way of thinking advocated by Turkle and Papert can only go so far in CS, and that it is inappropriate to couple a learning style with a gender stereotype.

On the other hand, constructivism—especially social constructivism—has much to say about the task of the teacher and the role of peers in education, and the theory can contribute to the analysis of the well-documented social difficulties faced by women in the CS classroom and laboratory.

Pedagogical Methods in CSE

Constructivism suggests that programming exercises should be delayed until class discussion has enabled the construction of a good model of the computer. Too often students become infatuated with the absolute ontology supplied by the computer; premature

attempts to write programs leads to bricolage and delays the development of viable models.

Group assignments and closed labs should be preferable to individual homework exercises, because they soften the brutality of the interaction with the computer and facilitate the social interaction that is apparently necessary for successful construction. Teachers and lab assistants should be trained to encourage *reflection* and to prevent the bricolage.

Implications for Research

The science-teaching literature shows that performance is no indication of understanding. CSE research like Madison's [9], which elicits the internal structures of the student, is far more helpful than research that measures performance alone and then draws conclusions on the success of a technique. A student's failure to construct a viable model is a failure of the educational process, even if the failure is not immediately apparent.

As computer literacy becomes common, if not universal, students will begin their academic studies with an effective model of a computer. However, I believe that research must be done to discover if these models are stepping-stones to the construction of effective models, or obstacles like naive physics.

Acknowledgements

I would like to thank Abraham Arcavi, Bat-Sheva Eylon and Yifat Ben-David Kolikant for their critiques of the first draft of this paper.

References

[1] R. B. Davis, C. A. Maher, and N. Noddings, editors. *Constructivist Views of the Teaching and Learning of Mathematics*. J. for Research in Mathematics Education. Monograph No. 4. National Council for the Teaching of Mathematics, 1990.

[2] A. A. diSessa, H. Abelson, and D. Ploger. An overview of boxer. *Journal of Mathematical Behavior*, 10:3–15, 1991.

[3] B. du Boulay. Some difficulties of learning to program. In Soloway and Spohrer [17], pages 283–299.

[4] R. Duit. Students' conceptual frameworks: consequences for learning science. In Glynn et al. [7], pages 65–85.

[5] P. Ernest. The one and the many. In Steffe and Gale [18], pages 459–486.

[6] S. M. Glynn. Explaining science concepts: a teaching-with-analogies model. In Glynn et al. [7], pages 219–240.

[7] S. M. Glynn, R. H. Yeany, and B. K. Britton, editors. *The Psychology of Learning Science*. Lawrence Erlbaum Associates, 1991.

[8] S. Holland, R. Griffiths, and M. Woodman. Avoiding object misconceptions. *SIGCSE Bulletin*, 29(1):131–134, 1997.

[9] S. K. Madison. *A Study of College Students' Construct of Parameter Passing: Implications for Instruction*. PhD thesis, U. of Wisconsin, 1995.

[10] R. E. Mayer, editor. *Teaching and Learning Computer Programming*. Lawrence Erlbaum Associates, 1988.

[11] R. D. Pea. Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research*, 2(1):25–36, 1986.

[12] D. Perkins, S. Schwartz, and R. Simmons. Instructional strategies for the problems of novice programmers. In Mayer [10], pages 153–178.

[13] L. Sherry. A model computer simulation as an epistemic game. *SIGCSE Bulletin*, 3(2):59–64, 1995.

[14] D. Sleeman, R. T. Putnam, J. A. Baxter, and L. Kuspa. An introductory pascal class: A case study of student errors. In Mayer [10], pages 237–257.

[15] D. Sleeman, R. T. Putnam, J. A. Baxter, and L. Kuspa. A summary of misconceptions of high school basic programmers. In Soloway and Spohrer [17], pages 301–314.

[16] J. P. Smith III, A. A. diSessa, and J. Roschelle. Misconceptions reconceived: A constructivist analysis of knowledge in transition. *The Journal of The Learning Sciences*, 3(2):115–163, 1993.

[17] E. Soloway and J. C. Spohrer, editors. *Studying the Novice Programmer*. Lawrence Erlbaum Associates, 1989.

[18] L. P. Steffe and J. Gale, editors. *Constructivism in Education*. Lawrence Erlbaum Associates, 1995.

[19] S. Turkle and S. Papert. Epistemological pluralism: Styles and cultures within the computer culture. *Signs: Journal of Women in Culture and Society*, 16(1):128–148, 1990.

[20] E. von Glaserseld. A constructivist approach to teaching. In Steffe and Gale [18], pages 3–15.