The Constructionist Role of Programming in the AI Era. Revisiting Seymour Papert's Teaching Children Thinking from 1970

Evolution of Constructionism: Past, Present, and Future

Ken Kahn[1], and Sylvia Libow Martinez[2]

[1] University of Oxford
[2] Constructing Modern Knowledge Press, Torrance, California, USA

Abstract

Long before constructionism had a name, Seymour Papert wrote about how a child could use a programming language to "learn to manipulate, to extend, to apply to projects, thereby gaining a greater and more articulate mastery of the world, a sense of the power of applied knowledge and a self-confidently realistic image of himself as an intellectual agent." He described programming languages as opening "a vast universe of things to do. But the real magic comes when this is combined with the conceptual power of theoretical ideas associated with computation." Here I argue that there is now another way to empower this kind of creativity and reflection. Constructionist learning can arise by conversing with a chatbot to build computer programs.

Keywords and Phrases: Seymour Papert, Programming languages for children, large language models, chatbots

1. Introduction

In 1970 Seymour Papert published *Teaching Children Thinking* where he articulated this vision:

> "Computation has had a profound impact by concretizing and elucidating many previously subtle concepts in psychology, linguistics, biology, and the foundations of logic and mathematics. I shall try to show how this elucidation can be projected back to the initial teaching of these concepts. By so doing much of what has been most perplexing to children is turned to transparent simplicity; much of what seemed most abstract and distant from the real world turns into concrete instruments familiarly employed to achieve personal goals." [4]

Chatbots are capable of matching this vision by providing age-appropriate explanations of the code they generate. They can ask students questions to help them understand the programs they are co-creating. Chatbots tirelessly and cheerfully answer student questions. While programming languages convey powerful ideas, they often

Kahn, K., & Libow Martinez, S. *Constructionism Conference Proceedings, 8/2025*, 413–417. https://doi.org/10.21240/constr/2025/20.X

burden students with mundane details. Chatbots can be instructed to hide the boring technical details and highlight the big ideas. And for those students who do want to master a programming language, chatbots can be good mentors. [2]

Applying computational ideas when learning "concepts in psychology, linguistics, biology, and the foundations of logic and mathematics" can lead to a deeper understanding of both the science and the computational ideas. I suspect that using chatbots one can do this more broadly and thoroughly than without them.

Those familiar with block-based languages such as Scratch [3] or Snap! [1] may be thinking "But there already are high-level programming languages whose syntax is so straight-forward that students can concentrate on higher-level issues." True, but there still are lots of details students need to master before doing ambitious projects. Consider just one example – numbers. When is arithmetic exact? Why does (1/2 + 1/3) + 1/6 equal 0.9999999999999999 while 1/2 + (1/3 + 1/6) equals 1? And large numbers behave oddly. 1e300 means 10^{300} while 1e400 is considered infinity. Other examples include variable scope confusion, coordinate systems, timing, and edge-of-screen behavior.

Chatbots can deal with these quirks and other details, allowing the student to focus on the design and the big picture. It is possible for students to acquire the same computationally powerful ideas associated with programming languages by interacting with chatbots in a thoughtful and inquisitive manner. However, many students may be motivated to build something they are passionate about as quickly as possible and only pay attention to the design of their creations. They may overlook the computational aspects, especially as chatbots become increasingly competent and rarely introduce bugs. Those students will still have many opportunities to improve their designs, critical thinking, and communication skills but might miss out on learning important computational concepts.

Interactions with chatbots can reinforce some powerful ideas even more effectively than programming alone. To effectively use chatbots, students need to hone their skills of clear communication of goals and actions. They need to provide constructive feedback to chatbots when they make mistakes or misunderstand.

Papert also wrote about how learning computational concepts may help in improving self-reflection about their own learning in general.

> "The most important (and surely controversial) component of this impact is on the child's ability to articulate the working of his own mind and particularly the interaction between himself and reality in the course of learning and thinking." [4]

An open question is how important is the formality of the descriptions of computational ideas to having this impact. Are informal natural-language based descriptions and experiences good enough for those students who don't try to master a programming language? A chatbot can be instructed to create runnable programs but only display to the learner the pseudocode equivalent in the desired natural language. The level of detail and the complexity of the language can be adapted to the needs of the learner. Pseudocode is optimally readable at the cost of being poorly suited for writing programs. But if the chatbot does the writing and the learner the reading, why expose the technical details and syntax of a programming language to the learner?

Perhaps chatbots can help students "articulate the working of [their] own mind." They can be customized to explain their reasoning. It is possible to ask a chatbot to simulate a conversation between different personalities as they write programs, thereby making the chatbot's reasoning visible. The recent reasoning models such as the "o" ChatGPT models can display their "chains of thought". This includes creating plans to break problems into parts, attempting and evaluating plan steps, and changing plans as problems are encountered. Maybe seeing descriptions of how the chatbots are thinking will help students become better at describing their own thinking.

Another way chatbots can help students become more reflective is by asking reflective questions at appropriate times. For example, a chatbot can be instructed to ask reflective questions after various versions of the program are created. And a chatbot can afterwards provide its own reflections on the learning experience.

My last quote from Papert discusses the role of large-scale projects and compares them to art classes where students may work on a piece over several weeks.

> "The similarity has several dimensions. The first is that the duration of the process is long enough for the child to become involved, to try several ideas, to have the experience of putting something of oneself in the final result, to compare one's work with that of other children, to discuss, to criticize and to be criticized on some other basis than 'right or wrong.' The point about criticism is related to a sense of creativity that is important in many ways ... – including, particularly, its role in helping the child develop a healthy self-image as an active intellectual agent." [4]

LLMs can provide an additional source of constructive feedback as projects progress. They are also great brainstorming partners. A large-scale project can involve dozens of exchanges. Ambitious projects should start small and be iteratively improved. I believe that a child will feel that the resulting app is their creation, not primarily the computer's.

However, I do worry as LLMs become more capable, the child's role may shrink. The day may come when a child can simply ask an LLM for an app and not need to have a long back and forth with the chatbot. Will students who give in to this temptation learn less? Will their self-image suffer? I don't think anyone knows the answer. But it is possible to avoid this by instructing the chatbot to assist only and not to produce ready solutions. At the minimum the student will be able to express their ideas as computationally sophisticated simulations, games, and other apps.

2. What other skills are learned?

I believe that understanding AI and effectively using generative AI are skills of growing importance. Many economists claim that people who use AI in their work are significantly more productive. As AI becomes more prevalent in the workplace, people's AI skills will be increasingly valued. And their value isn't only in the workplace or learning. People are finding AI helpful in planning trips, making financial decisions, cooking, and gardening among many other everyday tasks.

I believe that skills in working with AIs to create software can be applied to many other tasks. You learn how to express yourself, provide useful feedback, leverage AI's strengths, and mitigate its weaknesses.

3. Preliminary observations

I recently taught two 18-hour courses to students between 12 and 17 years old. Most of them had no programming experience and some had only a little experience using a laptop. I found that a very brief introduction to chatbots followed by a live demo creating two apps (fireworks and a riddling game) was sufficient for the students to begin to produce apps on their own. We alternated between my short lectures about additional capabilities (like adding images and sounds to apps), students taking turns presenting the progress they made since their previous presentation, and time to work on their apps while I was available to answer questions and help with problems. Frequently my answer to a question or problem was "Let's see what the chatbot has to say about this."

During the last hour, the students presented their favorite app or two (some had made as many as nine apps) to an audience. Some designed unique games, while others produced practical apps to maintain to-do lists and make study quizzes. As they demonstrated their creations, the students described the process of co-creation including obstacles surmounted.

I've also taught several online courses to students between 8 and 17 to create apps with chatbots. We met weekly for an hour for five weeks. I begin by introducing the basic ideas the first week and demoing a simple example. In the following weeks the students present their projects and discuss how they made their apps. Often problems come up that I could address directly but instead ask the student to tell the chatbot about the problem while everyone watches. This frequently resolves the problem.

I give the students freedom to choose their projects. I suggest ways they might improve their app (or more often ask them to ask a chatbot for suggestions). Some students improve their apps weekly while others come up with new ones every week. I have been pleasantly surprised how many impressive apps are created. Examples include apps that recommend novels, a water conservation game, a whack a mole game, a physic quiz, an image generator, a flower collecting game, a carbon footprint calculator, and an Indian history app.

Understanding how chatbots work is not necessary to use them effectively. Nonetheless I believe some part of the course should be devoted to how chatbots are trained and run. I don't want to take away much time from creative, reflective, and communicative activities so I limit this to less than one hour.

4. Conclusion

Conversing with chatbots is a new way of engaging in constructionist learning. Compared to computer programming without chatbot help, students can be more productive and focus on design and high-level issues. But the precision and details of computational ideas may be easy to lose.

Generative AI can help students create any kind of digital product by supporting debugging, iterative enhancement, and lowering the risks associated with exploring different alternatives. This includes co-authoring illustrated stories, simulating panels and debates, and creating text-based adventures.

At the current level of capability of chatbots, students are in control. But as they get better, they will be able to do more, leaving the students with fewer learning opportunities. Chatbots can, however, be prompted to refrain from doing so, leaving more of the design and planning to the students.

Referenced

Harvey, B., Garcia, D., Barnes, T., Titterton, N., Miller, D., Armendariz, D., McKinsey, J., Machardy, Z., Lemon, E., Morris, S., & Paley, A. (2014). Snap! (Build your own blocks). *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, Atlanta, GA, 749.

Kahn, K. (2025). *The Learner's Apprentice: AI and the Amplification of Human Creativity*. Constructing Modern Knowledge Press, Torrance, California. 2025

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), Article 16.

Papert, S. (1972). Teaching children thinking. *Programmed Learning and Educational Technology*, *9*(5), 245-255. Also presented at the Proceedings of IFIPS World Congress on Computers and Education, Amsterdam, The Netherlands, 1970