Computational X: Integrating Computational Thinking across Disciplines

Austin Cory Bart, Dennis Kafura, Clifford A. Shaffer, *Virginia Tech*

Abstract: Virginia Tech has recently added Computational Thinking as a specific requirement in its general education requirements and is seeking ways to integrate computing and other technology experiences into a wide variety of disciplines. What is "Computational Thinking", how do we teach it, and how do we know if our students have learned it? In this session, we will detail a definition of Computational Thinking, pedagogical and technological strategies for teaching it, and methods for assessing students' knowledge of the subject. Multiple contexts for teaching computing will be discussed, with particular attention given to our experiences in developing a Computational Thinking course for non-majors that is themed around real-world data science. This course integrates a number of modern pedagogical techniques including active learning and peer instruction, along with supporting technological innovations. The presentation will be suitable for educational researchers and instructors who might be unfamiliar with Computational Thinking, but are interested in how the topic can be integrated into their own courses and disciplines. Participants will be encouraged to discuss and plan concrete ways that their teaching and research can benefit from the subject.

Literature Review

Several formal definitions of Computational Thinking have emerged by professional organizations, including CSTA and Google - however, there is still discussion within the community about the exact meaning (Weinberg, 2007). Most sources seem to agree that Computational Thinking is more than just integrating technology and more than just knowing how to program; it represents a fundamentally different way of solving problems (Wing, 2006). Computational Thinkers are able to abstract complex reality into concrete representations, create algorithms to manipulate and refine those abstractions, and interpret the results in a meaningful way. Inherent to this domain are opportunities for students to be creative, expand their capabilities, and grapple with ethical concerns.

For many students, learning computation means learning an entirely new subject that blends language, logic, and quantitative skills (Weinberg, 2007). This means that learners will often have both conceptual challenges (Luxton, 2016) and motivational struggles (Shell, 2016). For the conceptual issues, instructors must be able to develop scaffolds, find appropriate metaphors, and help their learners develop a "Notational Machine" (Sorva, 2013) to model the state of programs. Meanwhile, in terms of motivation, students struggle with their self-efficacy and understanding the long-term career applications of computing. Situated Learning Theory provides a useful lens for distinguishing content (the material to be learned) and the context (the cultural milieu and anchoring that surrounds the learning experience), and encourages instructional designers to build experiences with a motivating community of practice in mind (Choi, 1995). A number of competing themes have emerged in the literature to contextualize learners' first computing experiences. Guzdial et al (2013) uses media creation and manipulation to provide opportunities for creativity, for instance, while our own approach (Kafura, 2015) incorporates real-world datasets that students use to answer questions.

Goals and Objectives

After this session, participants will be able to:

- Define Computational Thinking
- List and define critical elements of Computational Thinking
- Identify elements in their courses where Computational Thinking might be integrated
- List tools for teaching Computational Thinking
- Suggest ways that Computational Thinking might be assessed for their students

Description of Practice

The presenters have created an introductory course on Computational Thinking for non-Computer Science majors who have never programmed before. The course leverages a number of modern pedagogical techniques including active learning, situated learning experiences, and social learning theory. Considerable effort is spent on scaffolding tools for learners with potentially low self-efficacy. Theories of academic motivation guided the development of the course, so that students would be empowered to work with data that is useful to their long-term careers and interesting to them personally. Many course resources were refined through formal methods of Instructional Design, and are publicly available for instructors to adopt. These include an introductory block-based programming environment, an open-access eTextbook, and a collection of assignments and instructor materials. The course's primary assessment is designed as an authentic experience to answer a question from their own discipline by applying computational methods. This is paired with smaller assessments to measure their verbal knowledge and intellectual skill development with regards to tasks such as tracing code, interpreting code, and explaining core concepts of Computational Thinking.

Our course has been deployed for five semesters, and a large quantity of both quantitative and qualitative data has been collected on the learner's' experience. We will share our experiences with attendees and discuss what we think works best for introductory computing students. Beyond describing the presenters' own experiences teaching this course for over 5 offerings, there will also be a brief discussion of other approaches to integrating Computational Thinking across disciplines. This will include other course contexts (e.g., game design, media manipulation, and web design) and other forms of programming (e.g., using Excel or database software).

Discussion

Participants in this session will be encouraged to discuss and plan concrete ways to incorporate computing into their teaching and learning. Time will be dedicated to structured conversation for attendees to discuss in small groups about how Computational Thinking can connect to their discipline. There will also be discussion about how learners can better understand how computing is similar and different across contexts and fields. Finally, attendees interested in growing their Computational Thinking skills will be given references to a number of open-access, web resources including interactive textbooks and scaffolded digital learning environments.

References

Choi, J. I., & Hannafin, M. (1995). Situated cognition and learning environments: Roles, structures, and implications for design. Educational technology research and development, 43(2), 53-69.

Guzdial, M. (2013, August). Exploring hypotheses about media computation. In Proceedings of the ninth annual international ACM conference on International computing education research (pp. 19-26). ACM.

Kafura, D., Bart, A. C., & Chowdhury, B. (2015, June). Design and Preliminary Results From a Computational Thinking Course. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (pp. 63-68). ACM.

Luxton-Reilly, A. (2016, July). Learning to Program is Easy. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (pp. 284-289). ACM.

Shell, D. F., Soh, L. K., Flanigan, A. E., & Peteranetz, M. S. (2016, February). Students' Initial Course Motivation and Their Achievement and Retention in College CS1 Courses. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 639-644). ACM.

Sorva, J. (2013). Notional machines and introductory programming education. ACM Transactions on Computing Education (TOCE), 13(2), 8.

Weinberg, A. E. (2007). Computational thinking: an investigation of the existing scholarship and research (Doctoral dissertation, Colorado State University. Libraries).

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.