Computational Action in App Inventor: Developing Theoretical and Technological Frameworks for Collaboration and Empowerment

Mike Tissenbaum, University of Illinois at Urbana Champaign, miketissenbaum@gmail.com
Josh Sheldon, Massachusetts Institute of Technology, sheldon@mit.edu

Abstract: This special interactive session highlights the how the theory of computational action – which argues for an approach to computing education that is founded on the idea that, while learning about computing, young people should have the opportunity to do computing in ways that have direct impact on their lives and their communities – has informed the development of MIT's App Inventor, a blocks-based programming language for building native mobile apps. Using examples from our work and from partner institutions, we will show how App Inventor has enabled learners from around the world to collaboratively engage in computational action. With an eye towards furthering App Inventor's support of computational action, this session will also highlight: 1) The recently developed support for real-time collaboration between students working at different computers; and 2) How different App Inventor components allow students to engage in embodied, enactive, extended and embedded learning.

Introduction

Seymour Papert famously claimed that the computer is a "protean machine" capable of representing nearly any idea (1980), allowing students to creatively express themselves, develop their own voices, concretize their ideas, and develop diverse and innovative ways to build and learn (Blickstein, 2016). However, despite the considerable promise of computing, and computing education, to support students in developing and implementing their "big ideas", much of computing education's focus is on students learning "fundamentals" of programming (e.g., variables, loops, conditional, parallelism, operators, and data handling) (Wing, 2011; Brennan & Resnick, 2011). This focus on students learning the concepts and formalisms of computing, largely decontextualizes the computing they are learning from how students might use computing in their everyday lives. Papert argued that this focus on "skills and facts" works as a bias against the development of personally transformative and meaningful ideas (Papert, 2000). The push for computing education that focuses on the acquisition of computing "knowledge" detached from its real-world applications threatens to instill in young learners that computing is not relevant to them and not something they will need in their future lives or careers – an issue persistently found in math and physics education (Williams et al., 2003; Flegg et al., 2012). If we are serious about computing as a core literacy for students to be productive members of the 21st century society and workforce (NRC, 2013), we need to seriously rethink our core assumptions about computing education.

Across the educational landscape, there have been initiatives to situate student learning in authentic contexts. For instance, Project-Based Learning (Blumenfeld et al., 1991), which has students working collaboratively on personally relevant projects, has been shown to increase students' domain knowledge, intrinsic motivation, and a wide range of other critical 21st century skills. In contrast, many traditional approaches that situate computing in real-world contexts are often generic and impersonal (e.g., designing checkout systems for supermarkets) and fail to meaningfully to connect to the personal lives of students.

There are notable exceptions to this traditional approach. For instance, Scratch (scratch.mit.edu) has strived to make computing more personal, by allowing students to easily design and build games that can be shared with a worldwide community (Resnick et al., 2009). BlockyTalky (Shapiro et al., 2016), is a platform that allows learners to easily combine sensors and physical computing to develop their own musical instruments and other physical computing as a means of self-expression and exploration. An interesting similarity between these platforms is that they largely eschew the traditional text-based approach to programming, preferring instead a block-based metaphor in which segments of code click together in a puzzle-like format. This approach allows users to focus on their designing and building, rather than the messy grammar and syntax that have long been a barrier for engaging youth in computational practices (Maloney et al., 2004).

Another critical barrier in engaging youth in personally relevant computing education is the contexts in which they learn computing – often taking place in traditional computing labs, far removed from students' everyday lives (Klopfer, 2008). In order to truly make computing education meaningful for students and to connect to their lives outside the classroom, we need to develop educational solutions that transcend traditional classroom walls. Fortunately, the growing proliferation of mobile and ubiquitous computing can allow us to reconceptualize how and where students learn computing. By focusing on these devices, we can free students'

work from the desk-bound screen and connect it directly to their lives and communities. The use of mobile and ubiquitous technology as the platform students build on can open up new avenues for students to see their worlds as "possibility spaces" in which they can identify needs that are relevant to themselves and those in their communities and then build real solutions.

Computational action: Computing education for impact and empowerment

In order to effectively develop educational interventions that engage students in computing education that prioritizes personal relevance and the potential for impact, we need to reexamine the goals of CS education, particularly within K-12 settings. Critically, the goal of computing education needs to move beyond computational thinking towards a perspective of *computational action*. A computational action perspective on computing is founded on the idea that, while learning about computing, young people should have the opportunity to do computing in ways that have direct impact on their lives and their communities (Tissenbaum, et al., 2019). Two key dimensions underly the theory of computational action for supporting learning and development: (1) computational identity, and (2) digital empowerment. *Computational identity* builds on prior research that showed the importance of young people's development of scientific identity for future STEM growth (Maltese & Tai, 2010). For us, computational identity is a person's recognition of themselves as capable of designing and implementing computational solutions to self-identified problems or opportunities. Further, the students should see themselves as part of a larger community of computational creators. *Digital empowerment*, builds from the work of Freire (1990) which situates empowerment as the ability to critically engage in issues of concern to them, and Thomas and Velthouse (1990), who see empowerment connecting to the concepts of meaningfulness, competence, self-determination, and impact. As such, digital empowerment involves instilling in young learners the belief that they can put their computational identity into action in authentic and meaningful ways on issues that matter to them.

In order to develop computational action educational initiatives, we have developed a set of criteria that outline the key elements required (Tissenbaum, et al., 2019). *Supporting the formation of computational identity requires:* 1) Students must feel that they are responsible for articulating and designing their solutions, rather than working toward predetermined "right" answers. 2) Students need to feel that their work is authentic to the practices and products of broader computing and engineering communities. *Supporting the formation of digital empowerment requires:* 1) A significant number of activities and development should be situated in contexts that are authentic and personally relevant. 2) Students need to feel that their work has the potential to make an impact in their own lives or their community. 3) Students should feel that they are capable of pursuing new computational opportunities as a result of their current work.

MIT App Inventor: A platform for supporting computational action

The App Inventor platform

In order to realize the potential of computational action to move youth beyond Papert's vision of intellectually empowered students (1993) towards students who are empowered to change the world, we also needed to extend Papert's vision of bringing every learner into the computer lab (Papert, 1993; Klopfer, 2008) and instead move computing out into the world (Tissenbaum, Sheldon & Abelson, 2019). As a result, new platforms need to be developed that harness this potential and allowing students to focus on what they want to build and why (Lee et al., 2016), while simultaneously enabling them to quickly put their designs into action. In response, we have developed MIT App Inventor (appinventor.mit.edu), a blocks-based programming that allows anyone to develop fully-functional mobile applications.

App Inventor: Supporting collaboration both locally and globally

The myth of the lone programmer turning out the next great piece of software or app has largely been debunked (Fitzpatrick & Collins-Sussman, 2012). The reality is that effective programming is a collaborative pursuit that draws in many different skills and competencies. In educational settings, to support novice programmers to collaboratively develop their apps, several pedagogical approaches have been used. Perhaps most prominent is the approach of pair programming, in which students work side-by-side at a *single computer* continuously collaborating on the same code (Williams & Kessler, 2000). Pair programming has been shown to be effective in supporting students in engaging in collaborative discourse and learning from each other (Plonka et al., 2015), in addition to increasing student retention, confidence, and program quality (McDowell et al., 2006).

However, pair programming does have some notable constraints that can hinder its broad adoption. Perhaps the most critical is that both students need to be in the same classroom at the same computer. This can

limit the settings supported by pair programming and is particularly problematic if users are globally distributed. To support students who are developing apps to address global challenges, may require students to work with peers outside their own classroom. For instance, the global Technovation Challenge (technovationchallenge.org) has young women (12-18 years old) from all over the world working in distributed teams to develop apps using MIT App Inventor which address challenges both in both in their local communities and globally.

In response to the desire to enable members of its global community to effectively collaborate, the MIT App Inventor team is testing new features that allow users to collaboratively develop apps together in real-time. Similar to other widely used collaboration tools, like Google Docs, multiple users can synchronously see and edit the same app, with App Inventor updating in real-time to reflect the changes of all collaborators.

An additional challenge of pair programming is that it is designed to only support collaboration between two students, limiting the types of group configurations possible. With App Inventor's real-time collaboration functionality, there are no hard limits to the size of the group. With larger group sizes, students can work on separate parts of the app or work together to collaboratively build and debug portions of an app. Another possible configuration to explore is to have students break up into smaller pair programming teams within a larger group, allowing them to leverage the unique advantages of both pedagogical configurations. In recent work by the research team during a 12-week study at a large urban high school, we observed how the real-time collaboration system allowed student groups to transition between individual and collaborative problem-solving while working on collaboratively developed apps (Tissenbaum & Kumar, 2019).

Enabling 4E learning: Extending App Inventor into everyday objects and the world

By freeing computing education from the confines of the desktop and computer lab, App Inventor offers new opportunities for students to engage in 4E (embodied, enactive, extended, and embedded) learning. For instance, students used the newly integrated Bluetooth low energy (BLE) and Internet of Things (IoT) extensions to make apps that use light and moisture sensors to monitor and report on the health of plants (embedded); as well as, apps that use light and sound sensors along with embedded cameras to create security systems for their rooms (extended). Other students used motion and location sensors to help fishermen know when their traps get cut and track where they went, reducing waste and additional harm to the ecosystem (enactive). Another group combined interactive maps with geolocating and augmented interactive games to help users learn about and clean up the local river (embodied). Through these features, App Inventor allows students to develop new genres of apps that connect together and to the world in ways that resonate with the pedagogical goals of both computational action and 4E Learning.

Computational action + App Inventor: A harmony of theory and technology

MIT App Inventor was not originally built with computational action in mind. Instead, the theory of computational action arose as we observed how students interacted with it and the opportunities it afforded for students from all around the world to develop apps that could have real impact in their lives. It helped us realize we needed to move beyond teaching kids to *code* and instead *empower them to be problem solvers*. By allowing students to solve problems as they arise, rather than through canned exercises, we mirror Papert's own visions of how students should learn computing (Papert, 1996) and, perhaps more critically, the authentic ways of professionals (e.g., though sites like stackoverflow.com). Since its development (as shown in the examples above), computational action is now the underlying theoretical stance that drives the development and refinement of MIT App Inventor. Before a feature is released, we ask "How does this feature support learners to have an impact in their lives or those of their community? How does it reduce barriers for them to put their ideas into action? How does it support students to collaborate, share ideas, or contribute to the larger community of learners, creators, designers who use MIT App Inventor in classrooms, after school programs, makerspaces, and at home?" In this way, the MIT App Inventor team has been able to hold a lens up to its own development as well as to the apps users build, to refine our understandings of computational action, our methodical approaches for supporting it (e.g., materials and support tools), and approaches for measuring their efficacy.

References

Tissenbaum, M. & Kumar, V. (June, 2019). Seeing Collaboration Through the Code: Using Data Mining and CORDTRA Diagrams to Analyze Blocks-Based Programming. In *Proceedings of the 13th International Conference on Computer Supported Collaborative Learning*. Lyon, France.

Tissenbaum, M., Sheldon, J., & Ableson, H. (2019) From Computational Thinking to Computational Action. *Communications of the ACM*, *62*(3). 34-36

Blikstein, P. (2016). Travels in Troy with Freire: Technology as an agent of emancipation. *Educação e Pesquisa*, *42*(3), 837-856.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).

Fitzpatrick, B., & Collins-Sussman, B. (2012). *Team geek: a software developer's guide to working well with others*. O'Reilly Media, Inc.

Flegg, J., Mallet, D., & Lupton, M. (2012). Students' perceptions of the relevance of mathematics in engineering. *Intl. Journal of Mathematical Education in Science and Technology*, *43*(6), 717-732.

Klopfer, E. (2008). *Augmented learning: Research and design of mobile educational games*. MIT press.

Lee, C. H., & Soep, E. (2016). None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*, *49*(4), 480-492.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004, January). Scratch: a sneak preview [education]. In *Creating, connecting and collaborating through computing, 2004. Proceedings. Second International Conference on* (pp. 104-109). IEEE

Maltese, A. V., & Tai, R. H. (2010). Eyeballs in the fridge: Sources of early interest in science. *International Journal of Science Education*, *32*(5), 669-685.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, *49*(8), 90-95.

National Research Council. (2013). *Education for life and work: Developing transferable knowledge and skills in the 21st century*. National Academies Press.

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

Papert, S. (1993). The children's machine. *Technology Review.* Manchester NH, *96*, 28-28.

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

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. IBM Systems Journal, 39(3-4), 720-729.

Plonka, L., Sharp, H., Van der Linden, J., & Dittrich, Y. (2015). Knowledge transfer in pair programming: An in-depth analysis. *International journal of human-computer studies*, *73*, 66-78.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Shapiro, R. B., Kelly, A., Ahrens, M., & Fiebrink, R. (2016) BlockyTalky: A Physical and Distributed Computer Music Toolkit for Kids. *Proceedings of the 2016 conference on New Interfaces for Musical Expression*. Brisbane, Australia.

Thomas, K. W., & Velthouse, B. A. (1990). Cognitive elements of empowerment: An "interpretive" model of intrinsic task motivation. *Academy of management review*, 15(4), 666-681.

Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, *43*(5), 108-114.

Williams, C., Stanisstreet, M., Spall, K., Boyes, E., & Dickson, D. (2003). Why aren't secondary students interested in physics? *Physics Education*, *38*(4), 324.

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