Integrating Social Skills Practice with Computer Programming for Students on the Autism Spectrum

Kurt Eiselt
Department of Computer Science
University of California, Davis
Davis, CA, USA
eiselt@ucdavis.edu

Paul Carter
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
pcarter@cs.ubc.ca

***Abstract*—This Innovative Practice Work in Progress describes initial efforts in developing computer programming classes for students with high-functioning autism. From the students' perspective, technical skills appear to be the focus in these classes, yet we create opportunities for students to work on social skills.**

Keywords—autism, computer programming, social skills, communication

I. INTRODUCTION

The Centers for Disease Control and Prevention (CDC) estimate that 1 in 59 children who were eight years old in 2014 have autism [1]. Autism spectrum disorder (ASD) is a range of "complex neurodevelopment disorders characterized by repetitive and characteristic patterns of behavior and difficulties with social communication and interaction" [2]. As the word "spectrum" in "autism spectrum disorder" suggests, autism manifests itself in different ways. Some children with profound ASD exhibit multiple severe symptoms, while others with milder forms of ASD have subtle social deficits such as maintaining friendships and forming positive impressions with employers.

What is often overlooked in the on-going discussion of children with autism is the fact that these children grow up to be adults with autism, with the same impaired social interaction abilities. Nearly half of the children with ASD have average or above-average intelligence [3]. However, a recent study indicates that people with a diagnosis of high-functioning autism, even those with average intelligence, are no more likely to find a spouse or hold down a job than those with severe forms of autism [4].

In short, the long-term employment prospects for individuals with high-functioning autism appear to be no better than those for people with more disabling variants. Even those with above average intelligence who are unable to hold a job are likely to depend upon the care of others for their entire lives. One study indicates that the lifetime cost of providing care for someone with autism is $1.4 million (USD) when there is no intellectual disability, and $2.4 million when intellectual disability is involved. The yearly cost to the United States is estimated to be $236 billion [5]. Other studies report different dollar amounts [6], but in all cases the costs are overwhelming.

Some have suggested that computer science as a college major offers promising futures to young people on the autism spectrum [7]. Recognizing the potential in the ASD community, several companies are actively recruiting people with autism for programming and testing [8, 9, 10], and strive to provide these employees with a supportive accommodations in the workplace [11]. However, these companies assume that candidates for these positions have acquired the necessary social and computing skills elsewhere. Other organizations offer focused, immersive instruction in coding [12, 13], but these limited efforts are not likely to keep up with the demand for computing education from students with ASD. So, for the most part, the educational challenges fall on the traditional providers: colleges and universities (and a growing number of K-12 institutions) for computing education, and an army of dedicated, highly-trained therapists for social skills.

The same challenges that interfere with success in the workplace are likely to arise in school. For computer science educators wanting to maximize the chances of success for students with ASD, a search of the computer science education literature reveals a small but growing body of research into computing education for students with learning disabilities associated with autism and other disorders, but little investigation into merging social skills with that education [14, 15, 16, 17].

With the help of two speech-language pathologists, the authors of this paper have been exploring a merger of computing education and social skills education. We have prototyped computer programming classes in which, from the students' perspective, the technical component appeared to be the focus, yet we created situations for the students to work on understanding and fulfilling social expectations. Our initial goal is to demonstrate the feasibility of integrating communications practice with programming education and, we hope, to observe advancement in both skills. If the demonstration is successful, additional development could lead to classes that would provide students with sufficient computing skills to enable potential employers to see past the students' diagnoses, while at the same time enhancing social and communication skills that would make it easier for these students to succeed in the working world.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

II. The Students

Nine teens and pre-teens (ages 9-16) were invited to attend the first programming course. All students had a diagnosis of high-functioning autism spectrum disorder. All had expressed interest in computers or programming prior to the beginning of the classes, and all were able to bring their own laptop computers to class.

Eight students continued to the end of the first course, the ninth having dropped out early. Of those eight students, seven had attended weekly structured social groups sometime during the six months prior to the beginning of the programming class. These groups utilized the Social Behavior Mapping framework created by Michelle Garcia Winner [18] for improving individual social skills. Consequently, most of the students had been introduced to a common vocabulary for talking and thinking about social behavior, and this same vocabulary was used during the programming classes to cue desired social behavior.

III. The Programming Curriculum

A. The First Course: Visual Programming

The first course ran for eight weeks, with each class going for two hours per night, one night per week. We took a three-week break after the fifth class to accommodate external commitments on the part of the authors.

As we plotted the first course, we went forward with a few assumptions in mind:

- Though we want our students to gain competence with a text-based programming language, we should start with a visual programming language, as it might hold the attention of this particular group better than text. We chose Scratch.
- Everyone in this group plays computer games, so we had the students creating games as soon as we could.
- This group is not keen on listening to someone talk to them for minutes at a time; they would rather be doing something. We adopted an informal "I do, we do, you do" model: the programming instructor worked through a problem (the instructor's laptop display was projected on the wall), then the instructor and the students talked and coded through a related problem, and then the students worked through a third problem while the instructor and speech-language pathologist walked among them and provided assistance as needed.

The resulting course was structured as follows (many of our exercises were adapted from [19]):

- Week 1: Drawing shapes. Moving the sprite. Introduce the motion, events, control, sound, and pen palettes; coordinates; movement; degrees of rotation; loops.
- Week 2: Create an aquarium. Build on motion, coordinates, loops; add looks and sensing palettes; introduce animation through movement plus changing costumes: sense edges and reverse direction; sense sprite-to-sprite contact and take action using if- then.
- Week 3: Pong. This game has the same features (game loop, collision detection) as the Aquarium but adds user input; build on sensing, conditionals, motion, coordinates, loops; create a paddle that responds to a key press to move the paddle.
- Week 4: Telling a story through programming. Our client needs some animation to introduce herself at a conference talk; the animation must meet specific requirements (it is biographical and has a time limit); pairs of students work together on Scratch programs that tell the story of one segment of the client's life.
- Week 5: Hoppy Cat to Flappy Bird. A "midterm exam" of sorts; given the "Hoppy Cat" script as a starting point (the cat sprite hops over oncoming hurdles when a key is pressed), create the "Flappy Bird" script (the bird sprite navigates oncoming gates via key presses); work in pairs to adapt the existing script to the new task.
- Week 6: Whack-a-Mole. Build on and reinforce earlier ideas; introduce time for keeping track of how long the mole will appear on the screen; use random numbers to vary where the mole will appear on the screen; use mouse to position the mallet and "whack" the mole.
- Week 7: Space Invaders. Program adaptation; adapt the Pong paddle to be the Space Invaders missile launcher; start small then scale up; begin with one space invader that does not move; add movement to the single invader (traverses the screen while approaching the ground); introduce the cloning of sprites.
- Week 8: More Space Invaders. Add multiple rows of space invaders.

To maintain student engagement between classes, we gave a homework assignment at the end of each class. These assignments were typically to extend some feature of the program worked on in class. More details about the first course curriculum can be found in [20].

B. Social Strategies in the First Course

Ostensibly, the focus of this course was computer programming. We were equally concerned with the social aspects of our classes, and we were especially interested in using the computer programming classroom as a venue for our students to practice the social skills training they had received prior to the course (and, in many cases, continued to receive in parallel with the course). Specific social challenges we either anticipated or observed early on included:

- Following group instructions, especially once laptops were open and programming had begun.
- Working in groups, taking feedback from and giving feedback to peers, initiating comments or questions, working on common goals, and sharing responsibilities.
- Taking on the perspective of others (knowing what others are thinking about you, what kind of impression you're giving, and why this matters).
- Knowing how and when to ask for help.

To address these challenges, we implemented several strategies to bring social thinking to the forefront on occasion:

- With laptops closed, we began every class with a review of social expectations (e.g., look at the instructor, not your keyboard, when the instructor is talking), before there was any discussion of programming. This allowed us to use the vocabulary of the Social Behavior Mapping framework to remind the students of social expectations with small verbal (or sometimes visual) cues as needed during class.
- We used the Social Thinking Vocabulary [21, 22] to address any social difficulties that came up, helping to carry over learned concepts from previous social groups the students had attended.
- We promoted the idea that computer programming is often a collaborative effort by frequently employing exercises that required working together, giving insight to others, accepting feedback, and acknowledging the work of others. We gave direct instruction and feedback on how to do this when it was expected that the students were working together.
- We used "Pause" moments to have the students stop whatever they were doing and reflect on whether or not they were meeting social expectations (for example, "Were you thinking about what the instructor was saying? Were you talking to anyone about your ideas?"). This allowed for reflection by the whole group without the need to single out individual students.
- At the beginning of every session except the first, the students shared the results of their homework with the group. Toward the end of every session, we again held a "programming parade" in which each student demonstrated the day's accomplishments to their peers and, in some cases, to parents who had come to take them home. This provided structured time for students to ask questions and accept feedback. It also allowed us to change social expectations: at these times students would not be thinking about their own projects and would instead be thinking about others.

C. *The Second Course: Text-based Programming*

With an eye toward gradually preparing our students for college-level courses and, eventually, the workplace, we wanted to transition from visual blocks-based programming to the more traditional text-based programming. We also wanted to reduce the demand on the instructors, as the first course proved to be very labor-intensive. We invited all eight students to attend a second course, this time with Java programming on the agenda. This course was held several months after the first, and five of the eight students returned, the other three having departed because of a change in location.

This second course ran for a period of ten weeks and began with students working in the Greenfoot programming environment [23], which is designed to ease the learning of object-oriented programming concepts. Again, we met once each week for two hours at a time. Each Greenfoot session was driven by a worksheet that guided students through the steps of designing a new program. Starter code was supplied on a USB key at the start of class that provided a framework within which students developed their solutions. The week-by-week curriculum was as follows:

- Week 1: Introduction to Greenfoot environment. Acting with move and turn methods. Predicting the behavior of actors when modifications are made to code.
- Week 2: Programmatically adding actors to the world. Working with random values. Responding to keyboard input.
- Week 3: Working with different types of actors. Implementing the game of Pong.
- Week 4: Implementing a game of Hoppy Frog. Recall that students had seen a similar program in Scratch during their first course. Using random numbers to control frequency with which hurdles appear. Tracking a score.
- Week 5: Working with sound. Animating the size and transparency of a ball to accompany a sound that fades over time.

At this point, we took a multi-week break to accommodate conference travel and Spring break. During this time, the instructors regrouped and assessed how things were working. Our immediate observation was that the transition to text-based programming was not as gentle as we had hoped. We had spent a great deal of time in class translating error messages. Our students still enjoyed what they were learning and were not complaining, but their energy levels had seemed to decline over the five weeks. On the other hand, students and parents alike appreciated that they were learning a "real" programming language. We also noticed that interweaving individual and team based exercises was more successful than leaving the team based exercises to the end of the class.

D. *Re-Introducing Text-based Programming*

The complexity of our last project in Greenfoot was such that we felt that we had reached the limits of what our students could reasonably achieve. Although Greenfoot had served us well over the first part of the course, we decided to start over with a new programming language. We adopted Processing, which allows for the development of interactive, graphics applications. The language is built on top of Java but with the sharp edges rounded off. This allowed us to maintain our focus on learning a "real" programming language while working in a simple environment to develop fun, interactive applications.

At the start of each session, we briefly talked about the plan for the day, had the students read the relevant sections of *Learning Processing* by Daniel Shiffman [24], and then supported them through programming exercises. The book is presented in worksheet style and hence allowed for continuity in the format of the workshops, which we felt had been very successful during the first part of the course. This eased the workload on the instructors as we were able to provide a short handout that pointed students to chapters in the text and particular exercises that we wanted them to work through. The worksheets presented in Shiffman [24] included exercises that

were less prescriptive and allowed for more freedom in the programs that were designed. Students were also encouraged to come up with their own sketches and designs after class. Briefly, the remaining five classes revisited core concepts presented in the first part of the course as follows:

- Week 6: Introduction to the Processing environment. Shiffman, Chapter 1 and 2.
- Week 7: Dynamic sketches, interaction with mouse clicks and key presses. Shiffman, Chapter 3.
- Week 8: Variable declaration and initialization. Shiffman, Chapter 4.
- Week 9: Conditionals. Shiffman, Chapter 5.
- Week 10: Loops. Shiffman, Chapter 6.

IV. OBSERVATIONS

Although we put in a great deal of effort, at this time we have no real evidence of improvement in student learning or behavior. We can, however, offer some purely anecdotal observations that may, at the very least, encourage others to give some thought to the issues outlined in this paper.

We think it is safe to say that most of our students know more about computer programming than they did when they started. Many of our students had little to no programming experience when they began, but by the end of the first course they were able to read and write Scratch scripts (with varying degrees of confidence), and were conversant in fundamental programming concepts such as loops, conditionals, and variables. By the end of the second course, the remaining five students were able to read and write Processing programs.

As for social interaction, we believe that we saw several improvements. For example, we believe we observed an increase in spontaneous greetings to peers as they arrived for classes. We believe there was an increase in the students' ability to offer positive and constructive comments for others, and that they were better able to receive feedback. We believe that, by the end of the classes, they were better able to maintain conversations with their peers, especially when the conversations were about programming.

One principle we stressed often was "Programming to Specifications": if you are writing a program for someone else, write the program that was asked for, even though that is not the program that excites you the most. This is also a key principle in real-world software development, but our students found it to be quite challenging. People with ASD are thought to have difficulty seeing things from another person's perspective, or to imagine the thoughts and feelings of others [25]. Our students demonstrated a tendency to co-opt a programming exercise that we specified for them and make modifications that they wanted, without regard for the given specifications. The dilemma for us was to impress upon them the need to program to specifications without putting a damper on their desire to explore and experiment. Over time we learned to put specific boundaries on which components of the program were open to experimentation and which were not (i.e., "The space invaders may be anything you want them to be" versus "The space invaders must all move in the same direction at the same time"). We believe they now understand programming to specifications better than at the beginning of the course, although this may not be their preferred way of programming.

Despite the best efforts and intentions of our speech-language pathologists and ourselves, the team-based exercises weren't always successful. Often, students would begin a team exercise working together but quickly split up and continue on their own. The degree to which students were willing to work together may have been related to the nature of the exercise or to the relative skill sets of the students involved.

V. NEXT STEPS

One of the biggest lessons learned was to make communication a key component of the task at hand, whenever possible. One of our speech-language pathologists observed that the computer science instructors were too quick to provide direct answers to questions about programming. Given that one of our goals was for students to improve their communication and social skills, we needed to look for more opportunities for students to engage with each other. In asking questions of each other, they also acquired experience answering questions. As we gained more experience, we became better at inviting a student who had completed an exercise to help other students who were having difficulty, and observed some success with this strategy.

Through discussions with other professionals in the autism community, we were made aware of other opportunities for integrating communication skills in existing exercises. For example, in the Scratch aquarium exercise, we might have the inhabitants talk to each other, not just bounce off each other. Also, in the biography exercise, instead of animating someone else's life story, we could have the students animate their own story. It has also been suggested that we include neurotypical students, with similar technical skill levels, in the class. This would provide valuable opportunities to interact and work with people who are not on the spectrum, thereby modeling a more typical environment.

We are delighted to note that a group of undergraduate students at the University of British Columbia reached out to the authors for advice on offering coding workshops for students with ASD. They have since founded The C.O.D.E. Initiative, a not-for-profit organization that has now offered five coding workshops with approximately 50 students enrolled in total [26].

VI. CONCLUSION

As the Information Technology industry grows increasingly receptive to hiring from the autism spectrum, and other organizations recognize the need to provide spectrum-oriented computing education, the question of how best to deliver that education must be answered. Should social skills training take place independently from the computing education, or is there an advantage to students on the spectrum when the two educational efforts are merged? Our first small steps toward combining computer programming and social skills training suggest some benefits, and we plan to expand the investigation into how the intersection of these two ideas can influence development in both areas for these students.

ACKNOWLEDGMENT

Our thanks to the speech-language pathologists, Jenny Sojat and Claire Jones, whose expertise and guidance made this work possible, and to Adrienne Decker for her editorial assistance.

REFERENCES

[1] "Prevalence of autism spectrum disorder among children aged 8 years – Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2014," retrieved April 30, 2018, from Centers for Disease Control and Prevention: https://www.cdc.gov/mmwr/volumes/67/ss/ss6706a1.htm

[2] "Autism fact sheet," retrieved September 1, 2017, from National Institute of Neurological Disorders and Stroke: http://www.ninds.nih.gov/disorders/autism/detail_autism.htm

[3] "10 things to know about new autism data," retrieved February 18, 2015, from Centers for Disease Control and Prevention: http://www.cdc.gov/features/dsautismdata/

[4] M. Mordre, B. Groholt, A.K. Knudsen, E. Sponheim, A. Mykletun, and A.M. Myhre, "Is long-term prognosis for pervasive developmental disorder not otherwise specified different from prognosis for autistic disorder? Findings from a 30-year follow-up study," 2012. *Journal of Autism and Developmental Disorders*. 42 (June 2012), 920-928.

[5] "Lifetime costs of autism average $1.4 million to $2.4 million," retrieved February 18, 2015, from Autism Speaks: https://www.autismspeaks.org/science/science-news/lifetime-costs-autism-average-millions

[6] C. Dudley and J.C. Herbert, "The value of caregiver time: costs of support and care for individuals living with autism spectrum disorder," 2014. *The School of Public Policy: SPP Research Papers*. 7 (January 2014), Issue 1. Retrieved February 18, 2015, from the University of Calgary: http://www.policyschool.ucalgary.ca/sites/default/files/research/emery-autism- costs.pdf

[7] T. Grandin, "Choosing the right job," retrieved August 21, 2017, from the Autism Research Institute: https://www.autism.com/advocacy_grandin_job

[8] C. Bryant, "SAP seeks programmers with autism," May 21, 2013. Retrieved February 18, 2015, from Financial Times: http://www.ft.com/cms/s/0/2511f43a-c22b-11e2-8992-00144feab7de.html#axzz3S98xV7cJ

[9] B. Lam, "Why some companies are trying to hire more people on the autism spectrum," December 28, 2016. Retrieved August 21, 2017, from *The Atlantic*: https://www.theatlantic.com/business/archive/2016/12/autism-workplace/510959/

[10] M. Nickelsburg, "A year later, how Microsoft's jobs program for people with autism is working," September 6, 2016. Retrieved August 21, 2017, from GeekWire: https://www.geekwire.com/2016/year-later-microsofts-jobs- program-people-autism-working/

[11] M.R. Morris, A. Begel, and B. Wiedermann, "Understanding the challenges faced by neurodiverse software engineering employees: towards a more inclusive and productive technical workforce," 2015. ASSETS '15, October 26-28, 2015, Lisbon, Portugal.

[12] "Better futures for adults with autism," retrieved August 31, 2017, from nonPareil Institute: http://www.npitx.org

[13] "Coding/autism," retrieved September 1, 2017, from Coding/Autism: http://codingautism.com/about.html

[14] M.A.L. Egan, "Asperger's syndrome in the CS classroom," 2005. Proceedings of the 36th ACM Technical Symposium on Computer Science Education.

[15] J. Gribble, D. Harlow, A. Hansen, and D. Franklin, "Cracking the code: the impact of computer coding on the interactions of a child with autism," 2017. IDC '17, June 27-30, 2017, Stanford, CA, USA.

[16] M. Israel, Q.M. Wherfel, J. Pearson, S. Shebab, and T. Tapia, "Empowering K-12 students with disabilities to learn computational thinking and computer programming," 2015. *TEACHING Exceptional Children*, vol. 48, no. 1, pp. 45-53.

[17] S. Wille, J. Century, and M. Pike, "Exploratory research to expand opportunities in computer science for students with learning differences," 2017. Computing in Science and Engineering, vol. 19, no. 3, pp. 40-50.

[18] "Social thinking," retrieved February 18, 2015, from Social Thinking: https://www.socialthinking.com

[19] J.L Ford Jr. Scratch 2.0 Programming for Teens, 2nd ed. Cengage Learning PTR, Boston, 2014.

[20] K. Eiselt and J. Sojat, "Teaching programming on the autism spectrum: an experience report," 2015. Proceedings of the 20th Western Canadian Conference on Computing Education (WCCCE), Nanaimo, BC, Canada.

[21] M.G. Winner. Thinking about You, Thinking about Me, 2nd ed. Think Social Publishing, 2007.

[22] M.G Winner. Think Social! A Social Thinking Curriculum for School-Age Students. Think Social Publishing, 2008.

[23] M. Kölling. Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson Higher Education, 2009.

[24] D. Shiffman. Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction. Morgan Kaufmann, 2008.

[25] S. Baron-Cohen. Mindblindness: An Essay on Autism and Theory of Mind. MIT Press, 1995.

[26] S. Moreno-Garcia, "De'coding' autism stereotypes," retrieved March 22, 2018, from https://medium.com/ubcscience/decoding-stereotypes-81cd66ecefbf