Computational Thinking in University Mathematics Education: A Theoretical Framework

Chantal Buteau	Eric Muller	Joyce Mgombelo	Ana Isabel Sacristán
Brock University	Brock University	Brock University	Cinvestav-IPN

In recent years "Computational thinking" has become a trending topic among teachers who have seen their curricula include the term, and researchers who seek to pinpoint both what it means and how it can be implemented in a meaningful way in classrooms. We see a crucial need in mathematics education to understand how students could be empowered to participate in the computational thinking that is now becoming an integral part of the mathematics and broader community. In our research, we are interested in examining how university mathematics students may come to appropriate programming and engage in computational thinking for mathematics, as mathematicians would do. In this paper, we present the theoretical framework that grounds our research.

Keywords: computational thinking, instrumental genesis, programming, third pillar of scientific inquiry, undergraduate mathematics

Introduction

Before the advent of the personal computer, Papert (1971) envisioned a world in which children fluently program computers, using them as a tool to act as young mathematicians. Nearly half a century later, we've witnessed a widespread resurgence of interest in that vision, manifested in educational reforms (e.g., in Europe: Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016) and research regimes (e.g., Computational Thinking in Mathematics Education, n.d.) in the name of computational thinking, which now is deemed a 21st century skill. We see a crucial need to understand how students can be empowered to participate in such computational thinking that has become an integral part of the mathematics and broader community.

This paper focuses on the theoretical framework that grounds our recently launched 5-year study, funded by the Canadian Social Sciences and Humanities Research Council (SSHRC), that seeks to examine how postsecondary mathematics students learn to use programming as a computational thinking instrument for mathematics. It is a naturalistic study that takes place in a sequence of three programming-based mathematics courses implemented in the mathematics department at Brock University (Canada) since 2001, where undergraduate mathematics majors and future mathematics teachers learn to design, program, and use interactive computer environments to investigate mathematics conjectures, concepts, theorems, or real-world applications (Buteau, Muller, & Ralph, 2015; Muller, Buteau, Ralph, & Mgombelo, 2009). The objectives of our research include: (a) describing students' instrumental geneses of using programming as a computational thinking instrument for mathematics; (b) exploring whether or not students appropriated it and, if so, have sustained it beyond course requirements; and (c) identifying how instructors create a learning environment to support students' instrumental geneses. This study builds on our past and ongoing research (e.g., Buteau & Muller, 2014; Buteau, Muller, & Marshall, 2015; Buteau, Muller, Marshall, Sacristán, & Mgombelo, 2016).

Proposed Theoretical Framework

We start by discussing computational thinking and programming from a broad perspective based on the work of Wing (2008) and others. We then turn our attention to

computational thinking in mathematics portrayed by mathematicians' research practices—for example, as stated by the European Mathematical Society (2011). This leads to a discussion of computational thinking in mathematics education, which in turn is informed by the work of Weintrop et al. (2016) and the constructionist paradigm (Papert and Harel, 1991). Next, we elaborate on our view of learning mathematics by engaging in computational thinking drawing on some ideas from the work of Lave and Wenger (1991). Finally, we discuss Guin and Trouche's (1999) instrumental approach framework to inform our understanding of technology integration in mathematics teaching and learning.

Computational Thinking

Wing (2014) describes computational thinking as "the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out" (para. 5). Thus, computational thinking is an underlying process to computer programming. And as Grover and Pea (2013) state, computer programming "is not only a fundamental skill of [computer science] and a key tool for supporting the cognitive tasks involved in [computational thinking] but a demonstration of computational competencies as well" (p. 40). Wing (2008) explains that "the essence of computational thinking is abstraction" (p. 3717) and elaborates:

> Computational thinking is a kind of analytical thinking. It shares with mathematical thinking in the general ways in which we might approach solving a problem. It shares with engineering thinking in the general ways in which we might approach designing and evaluating a large, complex system that operates within the constraints of the real world. It shares with scientific thinking in the general ways in which we might approach understanding computability, intelligence, the mind and human behaviour. (p. 3717)

The relationship of computer programming and computational thinking with mathematical and scientific thinking and learning has been recognized since the development of the Logo programming language (cf., Papert, 1980a; Feurzeig & Lukas, 1972). This relationship is also highlighted in Brennan and Resnick's (2012) proposed three-dimensional framework characterizing "computational thinking" in terms of

> *computational concepts* (the concepts designers engage with as they program, such as iteration, parallelism, etc.), *computational practices* (the practices designers develop as they engage with the concepts, such as debugging projects or remixing others' work), and *computational perspectives* (the perspectives designers form about the world around them and about themselves). (p. 1)

In the following sections we discuss computational thinking in mathematics and computational thinking in mathematics education.

Computational Thinking in Mathematics

In terms of the development of mathematics itself, the European Mathematical Society (2011) recognized an emerging way of engaging in mathematical research: "Together with theory and experimentation, a third pillar of scientific inquiry of complex systems has emerged in the form of a combination of modeling, simulation, optimization and visualization" (p. 2). The notion of a third pillar had been raised previously in a 2005 report by the United States' President' Information Technology Advisory Committee (2005) highlighting the role of digital technology: "Together with theory and experimentation, computational science now constitutes the 'third pillar' of scientific inquiry, enabling

researchers to build and test models of complex phenomena" (p. 1). In 2016, mathematicians who led a 6-month long thematic semester on *Computational Mathematics in Emerging Applications* at the Centre de recherches mathématiques (CRM) in Montreal (Canada) indicated that:

> A fundamental change is taking place in the role of applied and computational mathematics. The relationship between the modelling, analysis, and solution of mathematical problems in applications has changed. … In emerging applications, the choice of models goes hand in hand with the computational tools and the mathematical analysis. (CRM, 2016, para. 1)

These emerging practices in mathematics research, we argue, fall under the umbrella of computational thinking for mathematics and are grounded on programming technology. Indeed, Weintrop et al.'s (2016) taxonomy (see Figure 1) gives insights into the computational thinking engagement by mathematicians and scientists, which encompasses the activities described by the European Mathematical Society (2011) and by the organizers of the computational mathematics session at CRM. Weintrop et al.'s work was based on an extensive literature review, an analysis of mathematics and science learning activities, and interviews with "biochemists, physicists, material engineers, astrophysicists, computer scientists, and biomedical engineers" (p. 134); the authors also outline what they believe to be the integral computational thinking practices for mathematics and science. Broley, Buteau, and Muller (2017) exemplified, through concrete research of mathematicians' work, the different forms of integral computational thinking practices proposed by Weintrop et al.

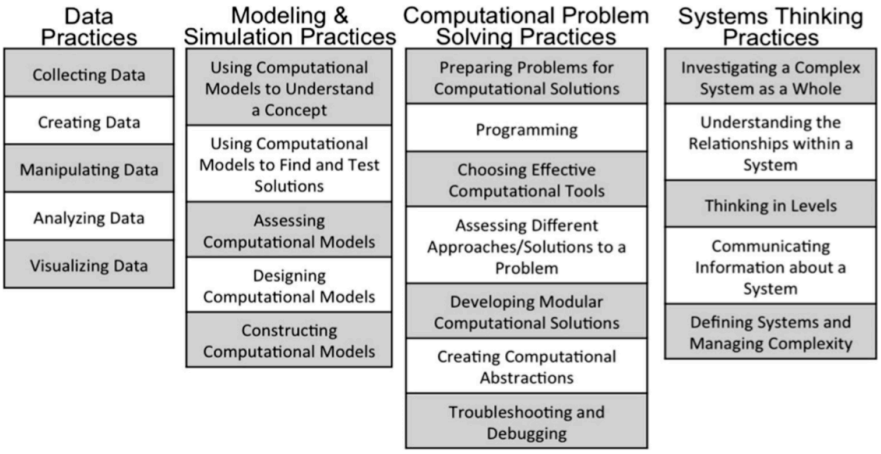

Figure 1. Taxonomy of computational thinking in mathematics and science (Weintrop et al., 2016, p. 135).

Adopting Brennan and Resnick's (2012) framework in the context of mathematics, the work of Weintrop et al. (2016) not only provides discipline-specific details for the computational practices dimension, but also foregrounds computational perspectives—that is, perspectives the mathematicians have come to recently develop about mathematics as a discipline "in line with the increasingly computational nature of modern science and mathematics" (Weintrop et al., 2016, p. 127).

Computational Thinking in Mathematics Education

Furthermore Weintrop et al. (2016) argue that "the varied and applied use of computational thinking by experts in the field provides a roadmap for what computational thinking instruction should include in the classroom" (p. 128). Their detailed taxonomy thus

provides us with what it means, in the mathematics classroom, to engage in computational thinking for mathematics as mathematicians would do.

As mentioned earlier, computational thinking in mathematics education has a legacy of over 45 years in the Logo programming language and in the theory of constructionism (Papert & Harel, 1991). The fundamental premise of the constructionist paradigm is to create student-centered learning situations for students to consciously engage in constructing (e.g., program) shareable, tangible objects, through meaningful —usually computer-based– projects: "People construct new knowledge with particular effectiveness when they are engaged in constructing personally meaningful products ... [that is] something meaningful to themselves and to others around them" (Kafai & Resnick, 1996, p. 214).

Studies of constructionism at higher-level mathematics education show how programming supports students' understanding of mathematical concepts (e.g., Leron & Dubinsky, 1995; Wilensky, 1995) and how it contributes to the development of critical thinking skills (e.g., Abrahamson, Berland, Shapiro, Unterman, & Wilensky, 2004; Marshall, 2012). In fact, Noss and Hoyles (1996) stress that a learner, when engaging in modifying a program, articulates relationships between concepts involved in a microworld "and it is in this process of articulation that a learner can create mathematics and simultaneously reveal this act of creation to an observer" (p. 54). In our work we concur with the constructionism approach for classroom implementation of programming and the computational thinking that it involves, and conceive mathematical learning by drawing from ideas found in situated learning theory, as described next.

Learning Mathematics by Engaging in Computational Thinking

Our view of learning draws from Lave and Wenger's (1991) work on communities of practice. Hoadley (2012) points that two definitions of community of practice stem from Lave and Wenger's work: (i) a feature based definition that derives from the words themselves meaning a community that shares practices and (ii) a process based definition which focuses on the process of learning whereby communities of practice are seen as groups in which a constant process of "legitimate peripheral participation" takes place. In our work, we rely on the process-based definition. Lave and Wenger use the concept of legitimate peripheral participation to describe how learners enter a community and gradually take up its practices. We use this idea of legitimate peripheral participation to understand how students learn mathematics through computational thinking. "Mathematics" is not seen as a body of knowledge to be acquired by the student, but rather as a process of participation through which the student gradually gains membership to a community (of mathematicians). Also, we do not see computational thinking from a cognitive point of view (e.g., seeing a computer as an interactive learning tool in illustrating concepts). Instead, we focus on how students create and use computer tools to engage in opportunities to participate peripherally in practices considered to be integral to the mathematical community as outlined by Weintrop et al. (2016). In other words, we focus on how students (newcomers) engage in computational thinking for mathematics as mathematicians (elders) would do.

This view on learning concords with the constructionism paradigm. Papert (1971) argued that "being a mathematician, … like being a poet, or a composer or an engineer, means *doing*, rather than knowing or understanding" (p.1), and that through programming mathematics, learners engage in "computational mathematics" (p.25) through which they mathematize. For Papert (1980b), the computer provides the learner a means for constructing "objects to think with" and "allow[s] a human learner to exercise particular powerful ideas or intellectual skills" (p.204) through exploration and discovery in a knowledge domain. This resonates with how many mathematicians and scientists use the computer in the 21st Century as described

earlier.

The work by Broley et al. (2017), cited earlier, exemplifies how undergraduate students learned mathematics through the construction of interactive computational objects (i.e., 'objects to think with'), and how these practices align with those of working mathematicians: for example, a first-year undergraduate's engagement in computational problem-solving practices –where she had to design, program, and use an interactive environment to explore, graphically and numerically, the behavior of a dynamical system based on a two-parameter cubic– shared similarities with a mathematician's engagement in his research on permutation of subsequences (see Figure 2).

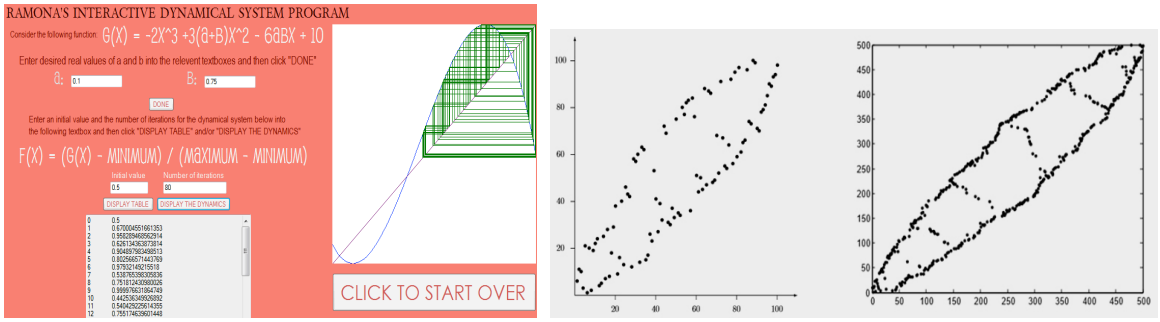

Figure 2. Examples of computational problem-solving practices. Left: screenshot of an undergraduate's exploratory work of a dynamical system. Right: screenshot of a mathematician's exploratory work on a permutation structure (Broley et al., 2017, pp. 4, 6).

When students become proficient at using programming to engage in computational thinking for mathematics "as mathematicians would do" (i.e., engaging in the computational practices as well as taking on the computational perspectives similar to how a mathematician would do), we consider that this technology has been integrated or that appropriation has occurred. We now turn to discussing this and how it can be assessed.

Students' Appropriation of Programming as a Computational Thinking Instrument

Cook, Smagorinsky, Fry, Konopak, and Moore (2002) explain that appropriation is a developmental process involving socially formulated, goal-directed, and tool-mediated actions through which learners actively adopt (i.e., what we could call "make their own") conceptual and practical tools, thus internalizing ways of thinking related to specific settings in which learning takes place. The instrumental approach (Rabardel, 1995/2002) is a useful framework for analyzing technological integration (Artigue, 2002; Guin & Trouche 1999) and gaining insights into how students appropriate a (technological) tool, and such an approach is used increasingly at the university level (cf., Gueude, Buteau, Mesa, & Misfeld, 2014).

The instrumental approach describes how artifacts (whether material or symbolic) are appropriated when they are transformed into instruments through schemes of usage and action by what is called *instrumental genesis* (Artigue, 2002). Trouche and Drijvers (2010) suggest that an instrument has been appropriated when a "meaningful relationship exists between the artifact and the user for a specific type of task" (p. 673). Thus, in order to assess the appropriation and technological integration, it is necessary to look at the instrumental genesis, by looking at both the artifact and its attached schemes. One way to do so is to look at the traces that students leave in their activity and what they do with an artifact (Trouche 2004). Parallel to this, it is also necessary to take into account the teacher's activity: his/her conceptions, design, and orchestrations of the teaching resources (Trouche, 2004) and the

instrumental integration, which is "how teachers organise the conditions for instrumental genesis of the technology proposed to the students and to what extent (s)he fosters mathematics learning through instrumental genesis" (Goos & Soury-Lavergne, 2010, p. 313). Instrumental integration describes four stages of growing technology use in the classroom (Assude, 2007): (a) instrumental initiation (stage 1)—students engage only in learning how to use the technology; (b) instrumental exploration (stage 2)—mathematics problems motivate students to further learn to use the technology; (c) instrumental reinforcement (stage 3)—students solve mathematics problems with the technology, but must extend their technology skills; and (d) instrumental symbiosis (stage 4)—students' fluency with technology scaffolds the mathematical task resulting in an improvement of both the students' technology skills and their mathematical understanding.

We associate these stages to a student's computational thinking development dimensions from Brennan and Resnick's (2012) framework: stages 1 and 2 to computational concepts, stages 2 to 4 to computational practices, and stages 3 and 4 to computational perspectives. And it is in stage 4 where we argue that the student has appropriated programming as an instrument for mathematics "as mathematicians would do" (both in terms of computational practices and perspectives) as mentioned in the previous section, which we term "programming as a computational thinking instrument for mathematics."

Next Steps for the Research

In this paper, we presented the theoretical framework underlying our study focused on how undergraduate mathematics students come to appropriate programming as a computational thinking instrument for mathematics. Brennan and Resnick (2012) suggest ways of assessing computational thinking development, including project portfolio analysis and interviews. Accordingly, in our research we will collect student participants' programming-based mathematics projects (14 in total over the three courses) together with their corresponding reflective journals, and students' lab reflections. We will also conduct semi-structured individual interviews with each of the participants in order to gain insights into students' creation process (including decision-making) and traces of their ongoing work. This is planned for two cohorts of 10 students each, followed over 3 consecutive years. Final interviews and questionnaires will be used at the end of the participants' 4- or 5-year program studies, to examine the sustainability of their programming use. Aligned with Trouche's (2004) recommendation, semi-structured interviews with course instructors, field notes of computer lab session observations, as well as course material will provide insights into the instructors' didactical aims and participants' learning environment. The latter data will also shed light on the instructors' pedagogical decisions and to what extent these are in accordance with the constructionist paradigm.

Acknowledgements

This study is funded by the Canadian Social Sciences and Humanities Research Council #435-2017-0367.

References

Abrahamson, D., Berland, M., Shapiro, B., Unterman, J., & Wilensky, U. (2004). Leveraging epistemological diversity through computer-based argumentation in the domain of probability. *For the Learning of Mathematics*, *26*(3), 19-45.

Artigue, M. (2002). Learning mathematics in a CAS environment: The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning*, *7*(3), 245-274.

Assude, T. (2007). Teachers' practices and degree of ICT integration. In D. Pitta-Pantazi & G. N. Philippou (Eds.), *Proceedings of the fifth congress of the European Society for Research in Mathematics Education* (pp. 1339-1348). Larnaka, Cyprus: Department of Education, University of Cyprus.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing computational thinking in compulsory education: Implications for policy and practice. *EU Science Hub*. Retrieved from https://ec.europa.eu/jrc/en/printpdf/175911

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the American Educational Research Association (AERA) annual conference.* Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Broley, L., Buteau, C., & Muller, E. (2017). *(Legitimate peripheral) computational thinking in mathematics*. Proceedings of the Congress of European Society for Research in Mathematics Education (CERME), Dublin (Ireland), February 2017.

Buteau, C., & Muller, E. (2014). Teaching roles in a technology intensive core undergraduate mathematics course. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: An international perspective on technology focused professional development* (pp. 163-185). Dordrecht, Netherlands: Springer.

Buteau, C., Muller, E., & Marshall, N. (2015). When a university mathematics department adopted course mathematics courses of unintentionally constructionist nature: Really? *Digital Experience in Mathematics Education*, *1*(2-3), 133-155. doi:10.1007/s40751-015-0009-x

Buteau, C., Muller, E., Marshall, N., Sacristán, A. I., & Mgombelo, J. (2016). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Digital Experience in Mathematics Education*, *2*(2), 142-156. doi:10.1007/s40751-016-0017-5

Buteau, C., Muller, E., & Ralph, B. (2015, June). Integration of programming in the undergraduate mathematics program at Brock University. In *Online Proceedings of Math+Coding Symposium*, London, ON. Retrieved from http://researchideas.ca/coding/docs/ButeauMullerRalph-Coding+MathProceedings-FINAL.pdf

Centre de recherches mathématiques. (2016). *Computational mathematics in emerging applications*. Retrieved from http://www.crm.umontreal.ca/act/theme/theme_2016_1_en/index.php

Computational Thinking in Mathematics Education. (n.d.). *About*. Retrieved from http://ctmath.ca/about/

Cook, L. S., Smagorinsky, P., Fry, P. G., Konopak, B., & Moore, C. (2002). Problems in developing a constructivist approach to teaching: One teacher's transition from teacher preparation to teaching. *The Elementary School Journal*, *102*(5), 389-413.

European Mathematical Society. (2011). *Position paper on the European Commission's contributions to European research.* Retrieved from http://ec.europa.eu/research/horizon2020/pdf/contributions/post/european_organisations/european_mathematical_society.pdf

Feurzeig, W., & Lukas, G. (1972). LOGO—A programming language for teaching mathematics. *Educational Technology*, *12*(3), 39-46.

Goos, M., & Soury-Lavergne, S. (2010). Teachers and teaching: Theoretical perspectives and classroom implementation. In C. Hoyles & J.-B. Lagrange (Eds.), *ICMI Study 17, technology revisited, ICMI study series* (pp. 311-328). New York, NY: Springer.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. doi:10.3102/0013189X12463051

Gueudet, G., Buteau, C., Mesa, V., & Misfeld, M. (2014). Instrumental and documentational approaches: From technology use to documentation systems in university mathematics education. *Research of Mathematics Education*, *16*(2), 139-155.

Guin, D., & Trouche, L. (1999). The complex process of converting tools into mathematical instruments. The case of calculators. *International Journal of Computers for Mathematical Learning*, *3*(3), 195-227.

Hoadley, C. (2012). What is a community of practice and how can we support it? In D. H. Jonassen & S. M. Land (Eds.), Theoretical foundations of learning environments (2nd Ed.) (287-300) New York: Routledge.

Kafai, Y. B., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Mahwah, NJ: Erlbaum, Routledge.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. New York, NY: Cambridge University Press.

Leron, U., & Dubinsky, E. (1995). An abstract algebra story. *American Mathematical Monthly*, *102*(3), 227-242.

Marshall, N. (2012). *Contextualizing the learning activity of designing and experimenting with interactive, dynamic mathematics exploratory objects* (Unpublished M.Sc. project report). Brock University, St.Catharines, ON.

Muller, E., Buteau, C., Ralph, B., & Mgombelo, J. (2009). Learning mathematics through the design and implementation of exploratory and learning objects. *International Journal for Technology in Mathematics Education*, *63*(2), 63-73.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers* (Vol. 17). Dordrecht, Netherlands: Kluwer.

Papert, S. (1971). Teaching children to be mathematicians vs. teaching about mathematics. *Artificial Intelligence Memo* No. 249. Retrieved from http://hdl.handle.net/1721.1/5837

Papert, S. (1980a). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Papert, S. (1980b). Computer-based microworlds as incubators for powerful ideas. In R. Taylor (Ed.), The computer in the school: tutor, tool, tutee (pp. 203–210). New York: Teacher's College Press.

Papert, S., & Harel, I. (1991). Situating constructionisn. In S. Papert & I. Harel (Eds.), *Constructionism* (pp. 1-12). Norwood, NJ: Ablex.

President's Information Technology Advisory Committee. (2005). *Computational science: Ensuring America's competitiveness*. Retrieved from https://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf

Rabardel, P. (1995/2002). *Les hommes et les technologies; approche cognitives des instruments contemporains.* Paris, France: Armand Colin.

Trouche, L. (2004). Managing complexity of human/machine interactions in computerized learning environments: Guiding students' command process through instrumental orchestrations. *International Journal of Computers for Mathematical Learning*, *9*, 281-307. doi:10.1007/s10758- 004-3468-5

Trouche, L., & Drijvers, P. (2010). Handheld technology for mathematics education: Flashback into the future. *ZDM: The International Journal on Mathematics Education*, *42*, 667-681. doi:10.1007/s11858-010-0269-2

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal for Science Education and Technology*, *25,* 127-147.

Wilensky, U. (1995). Paradox, programming and learning probability. *Journal of Mathematical Behavior*, *14*(2), 231-280.
Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A*, 366(1881), 3717-3725.
Wing, J. M. (2014, January 9). Computational thinking benefits society. *Social Issues in Computing,* 40th Anniversary Blog, University of Toronto. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html