Students Facing and Handling Challenges in Programming-Based Mathematics Inquiry Projects

Laura Broley
Brock University

Chantal Buteau
Brock University

Dorothy Levay
Brock University

Neil Marshall
Brock University

Eric Muller
Brock University

Jessica Sardella
Brock University

Joyce Mgombelo
Brock University

Ana Isabel Sacristán
Cinvestav

In this paper, we present an empirical study examining challenges undergraduate mathematics students (N=73) encounter when engaging in programming-based mathematics inquiry projects, and how they handle these challenges. Results suggest that a majority of students find the programming or the programming of mathematics the most challenging aspect of their engagement, for different reasons. Also, with more experience engaging in such projects, it seems students may become more able to handle their challenges in an independent manner (e.g., through reviewing concepts, persevering, or planning).

Keywords: Computing; Math Inquiry; Challenges; Student Learning; Instrumental Genesis.

There has been a growing interest in studying and implementing innovative approaches in undergraduate mathematics, such as inquiry-based mathematics education (Artigue & Blomhøj, 2013; Laursen & Rasmussen, 2019), in which students are invited to engage in the practices of professional mathematicians. There is also a recent push to integrate computer programming – or more broadly, computational thinking – in different subject areas and at all levels of education (Guzdial, 2019; Wiebe et al., 2020; Wing, 2014); though the potential of integrating programming in mathematics learning has been known for a long time (diSessa, 2018; Papert, 1980), including at the undergraduate level (e.g., Leron & Dubinsky, 1995; Wilensky, 1995). In particular, programming can support a certain inquiry-based approach, where students engage in computational practices used by some professional mathematicians (Weintrop et al., 2016).

Some undergraduate mathematics curricula have integrated programming in this sense. For example, at Carroll College in the United States, mathematics majors use programming as a problem-solving tool throughout their mandatory coursework (calculus, linear algebra, modelling, abstract algebra, etc.) and may eventually apply their programming skills in senior projects or theses (Cline et al., 2020). Another example, in the Canadian context, is a sequence of three courses called Mathematics Integrated with Computers and Applications (MICA), which have been implemented at Brock University since 2001. Throughout these courses, students engage in a sequence of 14 projects (including end-of-course projects on student selected topics), in which they design, program, and use interactive computer environments to investigate mathematics concepts, conjectures, theorems, or real-world situations (Buteau et al., 2015).

In their recent "Call for Research that Explores Relationships between Computing and Mathematical Thinking and Activity," to the international RUME community, Lockwood and Mørken (2021) argue that "serious consideration of machine-based computing [including programming] is largely absent from much of our research in undergraduate mathematics education" (p. 2). They suggest that much more needs to be investigated, "including identifying and exploring potential benefits, affordances, challenges, and problematic issues" (ibid.). They

also point out that the various approaches to integrating computing in university mathematics classrooms should provide opportunities for research. In our 5-year research study, we address the above gap, using the opportunity provided by the natural MICA environment. In particular, we examine the teaching and learning of using programming for engaging in pure or applied mathematical inquiry. As part of our study, we seek to better understand the challenges students face during their learning, which we see as potentially providing insights for implementation. In this paper, we present some initial findings addressing the following research questions: What challenges do undergraduate mathematics students encounter when engaging in programming-based mathematics inquiry projects? How do students explain their challenges? How do they handle their challenges? Are there any differences among different demographics of students?

Theoretical Framework

In our work, we frame students' engagement in programming-based mathematics inquiry projects using a development-process model (dp-model, Figure 1) proposed by Buteau and Muller (2010). According to the model, students' engagement involves different steps, which arise in a dynamic, non-linear fashion. For example, at Step 3, students design and program an "object" (or interactive environment) they will use for their inquiry, and this may occur in a cyclic manner with Step 4 (verification and validation of the programmed math). This model was developed through an analysis of MICA projects, a literature review (Marshall & Buteau, 2014), and analyses of student data (Buteau, Gueudet, et al. 2019). It has also been argued to align with the programming-based practices mathematicians use to conduct research – i.e., inquiry (Buteau, Gueudet, et al. 2019; based on Broley, 2015). Balt and Buteau (2020) provide a 5-minute video illustrating the model in the context of two selected pure and applied student inquiry projects.

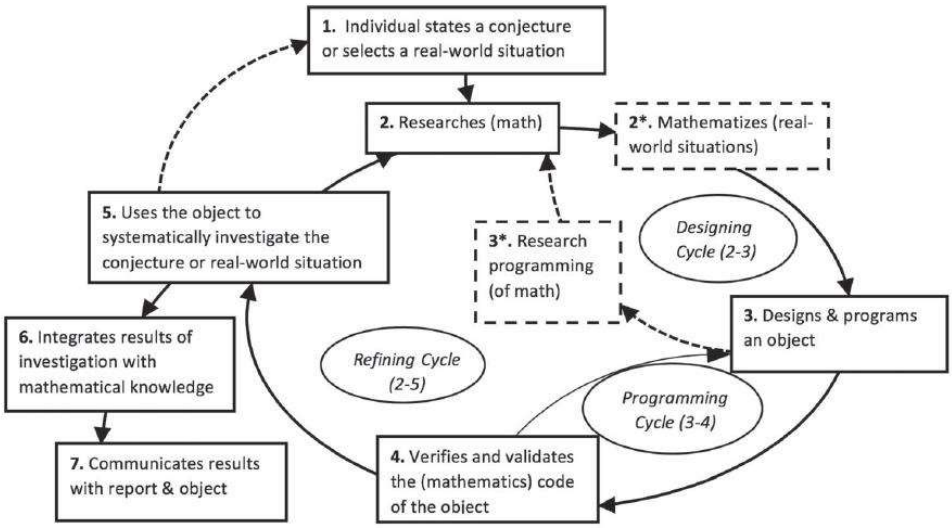

Figure 1. A model of students' engagement in mathematics inquiry projects (Buteau, Gueudet, et al., 2019, p. 6).

We further frame students' learning in the above model using the instrumental approach (Rabardel, 1995, 2002), whereby programming is an artefact (a human product) that may be transformed into a meaningful instrument (e.g., for conducting mathematical inquiry). This transformation – called instrumental genesis – involves the development of schemes (Vergnaud, 1998), including stable strategies (and principles underlying strategies) that enable the student to effectively achieve a goal (e.g., articulating a mathematical process in a programming language, as part of step 3 in the dp-model; Buteau, Gueudet, et al., 2019). In our case, a student's

instrumental genesis involves the development of a web of schemes (Buteau et al., 2020), i.e., interrelated schemes found at different steps of the dp-model.

Instrumental genesis is a complex process that can be challenging for students (Laborde, 2002). In our research, we have been documenting individual students' instrumental geneses through the sequence of inquiry projects offered in the MICA courses mentioned above, including mentioning some challenges that arose for those few individuals, in certain projects (Buteau, Gueudet, et al., 2019; Buteau, Muller, et al., 2019; Gueudet et al., 2020). In this paper, we move away from this in-depth approach and move to a wider student population to examine, more systematically, challenges (i.e., aspects of students' engagement that appear to cause the most issues, in terms of schemes, steps from the dp-model, or more general elements), and how students mostly handle these challenges (i.e., to move their mathematical inquiry forward).

Methods

As mentioned above, we work in the context of a sequence of three programming-based mathematics courses, MICA I-II-III, offered at Brock University, which engage mathematics majors and future mathematics teachers in 14 inquiry mathematics projects. Our study is part of a larger five-year naturalistic (i.e., not design-based) research aiming at understanding how students learn to use programming for authentic mathematical investigations, if and how their use is sustained over time, and how instructors support that learning.

As part of Years 2-4 of this research, all MICA students were invited to respond to an online questionnaire (~12-15 minutes long) at the end of their course (MICA I, II, or III). Participation was voluntary. Table 1 shows some relevant demographic information for the 73 participants. The questionnaire contained different sections: demographics; confidence in programming (for mathematics investigations); usefulness of programming; etc. In this paper, we focus on the questions asking 'what' students found most challenging in MICA projects, 'why', and 'how' they mostly handled it. Participants provided short written responses to each question.

Table 1. Information about participants that will be used in this paper.

Demographics				
Gender	MICA I	MICA II	MICA III	**Total**
Female	25	14	9	**48**
Male	14	7	4	**25**
Total	**39**	**21**	**13**	**73**

Each 'what', 'why', and 'how' response was coded, first individually (among 2-5 coders), and then through consolidation (which led to lists of 'codes'). Codes for the 'what' question were first grouped into 'themes'. Since each participant's responses to the 'why' and 'how' questions elaborated on their 'what' response, these codes were sorted first into groups by the 'what' themes, and then codes in each group were looked at and grouped into themes. This thematic regrouping was consolidated by 2 coders. Finally, we created different graphical representations of the frequency distribution of the results, including in relation to different demographic groups. We then interpreted our results using the theoretical frame elaborated in the previous section.

Since participation was voluntary, we cannot claim that our sample is representative of all MICA students. This is a limitation of our study.

Results

We present selected results in four sections aligning with our four research questions.

What students find most challenging

Our analysis led to eight themes, with distribution across participants given in Figure 2.

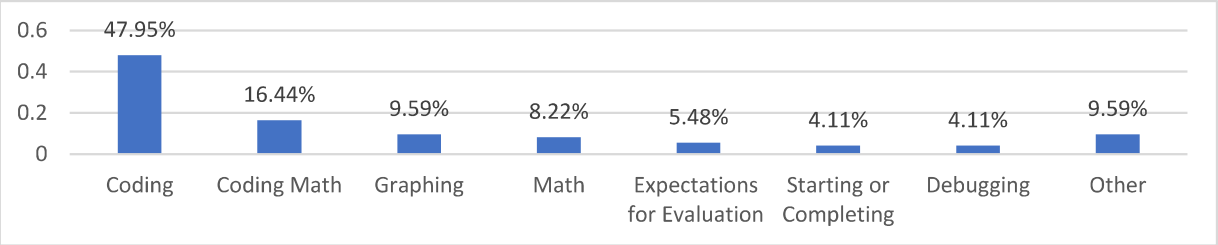

Figure 2. What participants (N=73) found most challenging about the MICA inquiry projects.

The part of the programming-based mathematics inquiry that most participants found most challenging was *coding* (e.g., "anything that requires a loop, an array, or long segments of code"), followed by *coding mathematical processes, including translating mathematics to code* (e.g., "making the computer do the math"). Both of these relate to students' (engagement with the) activity in steps 3-4 of the dp-model (i.e., the programming cycle). The other most challenging parts were mentioned in smaller proportion (at most 10%). Some are related to the programming cycle, such as *graphing and graphics*[1] (e.g., "getting graphs and picture boxes to do things") and *debugging* (e.g., "overcoming a roadblock or mistake in the coding"), making the issues at the programming cycle the most dominant ones (78% in total). All other challenges appear to relate to more general aspects of students' engagement (i.e., not necessarily related to a specific step in the dp-model): the *mathematics* (e.g., "understanding the math components"), *starting or completing the inquiry projects* (e.g., "at the beginning and setting up the program"), *expectations for evaluation* (e.g., "understanding what the professor expects"), and *other responses* such as "the imaginative process."

When reviewing participants' responses to both the 'what' and 'why' questions, we find that some of the latter challenges may be related to several steps in the dp-model. For example, the student mentioning the imaginative process explained: "I've always struggled with feeling confident in my own ideas ... coming up with my own conjecture …[was] the hardest for me," which we associate to step 1. Another student in the "other" theme specified: "I had a hard time understanding the content of the course in terms of how to apply my program to the questions asked of me," which we associate to step 5. And a student included in "math" said: "I found the written reports to be the most challenging … it just seemed to require more in depth thinking, rather than just knowing the math and programming it," which we associate to steps 6-7.

Why students find 'coding' and 'coding of mathematical processes' to be most challenging

We now consider the reasons behind the two greatest challenges reported by the highest proportions of participants. Our analysis of 46 participants' responses led to 11 themes, which can be categorized according to 4 different aspects of instrumental genesis.

The most dominant reason (32.6%) was *struggling with knowing what to do while coding, or starting or getting stuck with coding* (e.g., "I find it difficult to start as sometimes I don't know how to approach the math in a programming sense"). We infer that these students were

[1] Programming graphs in VB.Net requires coding a change of coordinate system.

struggling to mobilize or develop certain schemes (e.g., for articulating a math process in the programming language). Another reason, *struggling with making efficient code* (6.5%), also points to struggling to develop a certain scheme (e.g., "It's one thing to get the code to do what you want, it's another to get it to do that as efficiently as possible"). Thus, 39.1% of participants seem to suggest the development of certain schemes as causing the 'coding (math)' challenge.

Some reasons (with proportions totaling 36.9%) relate more to the nature of instrumental genesis: a process that takes time, effort, and confidence. This includes *coding (math) is new* (15.2%, the second most common reason), *needing more guidance* (10.9%, e.g., "I am expected to know commands and what to write with only being taught the basics"), *not being confident in coding* (6.5%, e.g., "I am not totally confident as a programmer"), and *needing more time* (4.3%, e.g., "It's a hard concept to understand in a short amount of time").

Two reasons (totaling 19.5%) point to the nature of the artefact (in this case, programming). This includes the third most common reason, *coding is finicky or picky* (13%, e.g., "one small thing can throw the whole code off") and *struggling to understand how a program works* (6.5%, e.g., "I really struggle to visualize how the program handles these codes and tend not to understand … how it is getting to the answer").

Finally, two reasons (totaling 8.6%) concern the nature of the mathematics: *finding it difficult to think computationally* (4.3%, e.g., "It generally takes a difficult path to go through the math and make it work") and *struggling with coding complicated or brand-new math* (4.3%, e.g., "some of the math is complicated enough on paper let alone telling a computer how to do it").

How students mostly handle what they find to be most challenging

Looking at how all 73 participants said they handled their most challenging part of their inquiry projects, 9 themes were found (summarized in Figure 3, left).

53.4% of participants indicated *seeking help from the instructor, teaching assistants (TAs), and/or peers* (e.g., "Asking as many questions, and getting as much help from the TA/Instructor as possible"; "I speak with peers and the professor in order to get on the right track").

Some other strategies (with proportions totaling 35.6%) include ways of independently gaining the knowledge required to address the challenge: *researching beyond my lecture material* (e.g., "Using the textbook and online sources to self-teach"), *reviewing my (previous) lecture material* (e.g., "sit down with my lecture notes and review them"), and *seeking a deeper understanding* (e.g., "just trying to understand the topics more deeply").

Two other strategies (totaling 21.9%) are associated with taking the time needed to adapt or develop one's schemes for the given situation: *persevering* (e.g., "Just testing new ideas until one works") and *taking my time* (e.g., "I took things one step at a time").

One strategy is specific to 'coding (math)' (4.1%): *planning before coding* (e.g., "trying to structure out some sort of design before writing programs"), which we infer as students developing part of a scheme for articulating a (math) process in the programming language.

A very small proportion (1.4%) indicated having *no strategy* (e.g., "I haven't [handled it]").

There were also 28.8% of *other strategies* mentioned, such as "I've been handling it by staying on top of my academics." Many of these could be connected through a larger category of *working through it on my own*, as could most of the themes in Figure 3, left (excluding *seeking help* and *no strategy*). We thus regrouped students according to their ability to work independently on programming-based mathematics inquiry, as summarized in Figure 3, right: 32.9% of students mostly *just seek help from others*, 38.4% mostly *just work through it on their own*, 23.3% *do both*, and 5.5% mostly *just struggle and/or have no strategy*. We interpret

students using a strategy including working through it on their own (61.7%) as suggesting that they may be "further along" in their instrumental genesis or more able to facilitate it themselves.

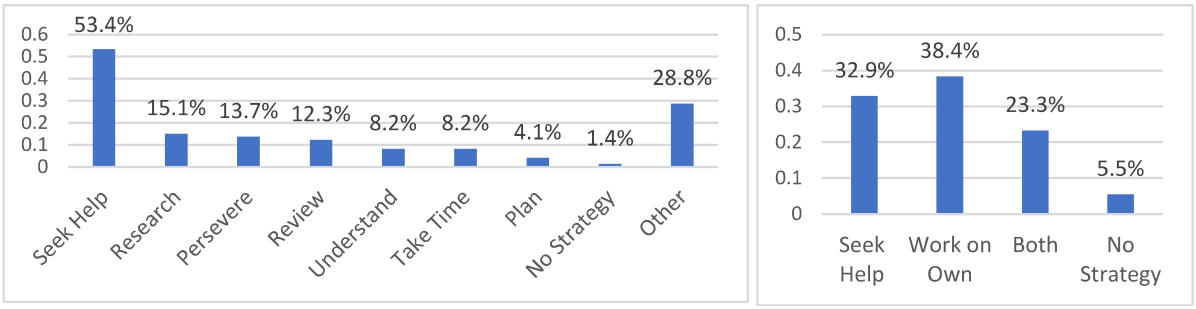

Figure 3. How participants (N=73) mostly handled their most challenging part of the MICA inquiry projects.

Some demographic comparisons in the 'what' and the 'how'

Finally, we present some results considering different demographic categories: year of study (MICA I vs. 'upper MICA,' i.e., MICA II or III) and gender (female vs. male).

When comparing MICA I and upper MICA participants, we found no clear difference in what was found to be the most challenging part of the inquiry projects. As for the way they mostly handled challenges, a greater proportion of MICA I participants indicated solely seeking help (38.5% vs. 26.5%), while a greater proportion of upper MICA participants indicated solely working through it on their own (47.1% vs. 30.8%); and a similar proportion of MICA I and upper MICA participants indicated utilizing both strategies. This may suggest that upper MICA students are more comfortable working independently in programming-based mathematics inquiry projects and they may be further along in their instrumental genesis.

There also seems to be no clear difference between what females and males find most challenging about MICA inquiry projects, except maybe for a slightly larger proportion of females pointing to the coding of mathematics (21% vs. 8%). Female and male participants also reported similar strategies for handling their most challenging part, in terms of solely working through it on their own or seeking help, though female students appear to do both slightly more (27% vs. 16%) and male students may be more likely to have no strategy (12% vs 2%).

Discussion

With this paper, we seek to answer Lockwood and Mørken's (2021) call for research by starting to examine the challenges students face when computing (including programming) is integrated in undergraduate math education, specifically in the context of mathematics inquiry.

In our study, student participants indicated in greatest proportion (~3/4) that the most challenging part of their inquiry is related to what we call "the programming cycle," which involves the design and creation of a program for the purposes of the inquiry, including the intertwined processes of translating mathematical processes into algorithms and validating the programmed mathematics. This aligns with work in computing education that has documented the many difficulties and misconceptions that novice programmers may have (Qian & Lehman, 2017): e.g., "their lack of well-established strategies and patterns often leads to various challenges in planning, writing, and debugging programs" (p. 6). The participants in our study pointed to different factors that may be contributing to their "coding (math)" challenge: not only the inadequacy of their current patterns of doing (which we framed in terms of schemes), but also the nature of the process required to develop such patterns (e.g., it takes time; Laborde, 2002). Participants also pointed to the nature of programming as a tool and the mathematics

involved in their programming-based mathematics inquiry. It is possible that the "newness" of this experience is underlying participants' responses, in which case we may expect different results in the future if students start learning programming in STEM subjects from a young age (cf. Wiebe et al., 2020). We note, nevertheless, that students gaining fluency in programming need not imply their fluency in using programming to conduct mathematical inquiry (Buteau et al., 2019). Moreover, the latter will always be inherently challenging: "the inquiry process develops as interplay between known and unknown in situations where some individual or group of individuals is *faced with a challenge*" (Artigue & Blomhøj, 2013, p. 798-799, our emphasis).

Work in computing education emphasizes that challenges may inhibit students' ability to learn and make progress (Qian & Lehman, 2017). In contrast, our study found that just because a student finds something "most challenging" does not mean that they struggle: participants described several strategies for handling their challenges, with only a few suggesting that they had no strategy. The high proportion of students indicating that they "seek help" aligns with principles of inquiry-based mathematics education. Laursen and Rasmussen (2019) identify four pillars of such an approach, which highlight the importance of students collaboratively processing mathematical ideas and instructors facilitating students' thinking in equitable manners (e.g., encouraging students to explain their ideas and ask for others' explanations and help). Importantly, this also aligns with how most professional mathematicians do their work: i.e., in cooperation, collaboration, or consultation with one another (e.g., Bass, 2011; Burton, 2004). This said, our study also found that students seem to develop more independent strategies as they progress in the MICA courses. We conjecture that students may be learning when to collaborate and when to work on their own. One limitation of the current study is that we did not distinguish between "seeking help" and "collaboration." Moreover, we only considered how students *mostly* face their *most* challenging part of their inquiry. Looking more into students' interactions with peers, instructors, and other resources could be an interesting direction for future work.

In their call for research, Lockwood and Mørken (2021) mention equity issues as one of four key research foci for the RUME community investigating computing, concluding that a potential downside to integrating computing in undergraduate mathematics is that it could perpetuate inequities in certain populations. Indeed, research in computing education has highlighted the greater challenges typically faced by female undergraduates (e.g., Margolis & Fisher, 2003). The small differences we observed among female and male participants seem to be promising and is consistent with some of our past work (Buteau et al., 2014). It is possible that this is connected to the inquiry approach taken in MICA courses. For instance, Laursen and Rasmussen (2019) indicate that "current studies show that inquiry classrooms can level the playing field for women … and argue why this may occur … but also show that this is not automatic" (p. 138). Looking further into which aspects of the MICA environment may support equitable outcomes could be another pertinent direction for future work.

Overall, our study seems to suggest that the teaching approach in the MICA courses may positively support students in learning to engage in programming-based mathematics inquiry. To think about implications for teaching, we need to examine more closely the teaching that the students actually receive. In particular, there is the question of what instructors can do to best support their students in handling their challenges.

Acknowledgments

This work is funded by the Social Sciences and Humanities Research Council of Canada (#435-2017-0367) and received ethics clearance (REB #17-088). We thank all our research assistants, in particular Nina Krajisnik and Kelsea Balt, for their work on the questionnaire data.

References

Artigue, M., & Blomhøj., M. (2013). Conceptualizing inquiry-based education in mathematics. *ZDM Mathematics Education, 45*, 797–810.

Balt, K., & Buteau, C. (2020, September 4). *Mandelbrot/rain design illustration: Using programming for pure/applied mathematics investigation* [Video]. YouTube. https://youtu.be/irTlCE-eXhc

Broley, L. (2015). *La programmation informatique dans la recherche et la formation en mathématiques au niveau universitaire* [Master's thesis, Université de Montréal]. Papyrus: Institutional Repository.

Bass, H. (2011). A vignette of doing mathematics: A meta-cognitive tour of the production of some elementary mathematics. *The Montana Mathematics Enthusiast, 8*(1&2), 3–34.

Burton, L. (2004). *Mathematicians as enquirers: Learning about learning mathematics.* Norwell, MA: Kluwer Academic Publishers.

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. I. (2019). University students turning computer programming into an instrument for "authentic" mathematical work. *International Journal of Mathematical Education in Science and Technology, 57*(7), 1020–1041.

Buteau, C., Gueudet, G., Dreise, K., Muller, E., Mgombelo, J., & Sacristán, A. (2020). A student's complex structure of schemes development for authentic programming-based mathematical investigation projects. In T. Hausberger, M. Bosch, & F. Chellougui (Eds.), *INDRUM2020 Proceedings* (pp. 513–514). University of Carthage and INDRUM.

Buteau, C., & Muller, E. (2010). Student development process of designing and implementing exploratory and learning objects. In V. Durand-Guerrier, S. Soury-Lavergne, & F. Arzarello (Eds.), *Proceedings of the sixth congress of the European Society for research in mathematics education* (pp. 1111–1120). Services des publications, INRP.

Buteau, C., Muller, E., Dreise, K., Mgombelo, J., & Sacristán, A.I. (2019). Students' process and strategies as they program for mathematical investigations and applications. In U.T. Jankvist, M. Heuvel-Panhuizen, & M. Veldhuis (Eds.), *Proceedings of the eleventh congress of the European Society for research in mathematics education* (pp. 2796–2803). Freudenthal Group & Freudenthal Institute, Utrecht University, and ERME.

Buteau, C., Muller, E., & Ralph, B. (2015, June 19-21). *Integration of programming in the undergraduate mathematics program at Brock University* [Paper presentation]. Math + Coding Symposium, London, ON, Canada. https://researchideas.ca/coding/docs/ButeauMullerRalph-Coding+MathProceedings-FINAL.pdf

Cline, K., Fasteen, J., Francis, A., Sullivan E., & Wendt T. (2020). Integrating programming across the undergraduate mathematics curriculum. *PRIMUS, 30*(7), 735–749.

diSessa, A.A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning, 20*(1), 3–31.

Gueudet, G., Buteau, C., Muller, E., Mgombelo, J., & Sacristán, A. (2020). Programming as an artefact: What do we learn about university students' activity? In T. Hausberger, M. Bosch, & F. Chellougui (Eds.), *INDRUM2020 Proceedings* (pp. 443–452). University of Carthage and INDRUM.

Guzdial, M. (2019). Computing for other disciplines. In S. Fincher, & A. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 584–605). Cambridge Handbooks in Psychology.

Laborde, C. (2002). Integration of technology in the design of geometry tasks with Cabri-Geometry. *International Journal of Computers for Mathematical Learning*, *6*(3), 283–317.
Laursen, S.L., & Rasmussen, C. (2019). I on the prize: Inquiry approaches in undergraduate mathematics. *International Journal of Research in Undergraduate Mathematics Education, 5*, 129–146.
Leron, U., & Dubinsky, E. (1995). An abstract algebra story. *American Mathematical Monthly, 102*(3), 227–242.
Lockwood, E., & Mørken, K. (2021). A call for research that explores relationships between computing and mathematical thinking and activity in RUME. *International Journal of Research in Undergraduate Mathematics Education*, 1–13.
Margolis, J., & Fisher, A. (2003). *Unlocking the Clubhouse: Women in Computing*. Cambridge, MA, USA: MIT Press, 2003.
Marshall, N., & Buteau, C. (2014). Learning by designing and experimenting with interactive, dynamic mathematics exploratory objects. *International Journal for Technology in Mathematics Education, 21*(2), 49–64.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NewYork, NY: Basic Books.
Qian, Y., & Lehman, J. (2017). Students' misconceptions an other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education, 18*(1), 1–24.
Rabardel, P. (1995). *Les hommes et les technologies: Approche cognitive des instruments contemporains*. Armand Colin.
Rabardel, P. (2002). *People and technology: A cognitive approach to contemporary instruments* (H. Wood, Trans.). Université Paris 8.
Vergnaud, G. (1998). Towards a cognitive theory of practice. In A. Sierpinska, & J. Kilpatrick (Eds.), *Mathematics education as a research domain: A search for identity* (pp. 227–240). Springer.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*, 127–147.
Wiebe, E., Kite, V., & Park, S. (2020). Integrating computational thinking in STEM. In C. C. Johnson, M. J. Mohr-Schroeder, T. J. Moore, & L. D. English, *Handbook of Research on STEM Education* (pp. 196–209). Routledge.
Wilensky, U. (1995). Paradox, programming and learning probability. *Journal of Mathematical Behavior, 14*(2), 253–280.
Wing, J. M. (2014, January 10). Computational thinking benefits society. *Social Issues in Computing*. http://socialissues.cs.toronto.edu/index.html%3Fp=279.html