Students' Process and Strategies as They Program for Mathematical Investigations and Applications

Chantal Buteau[1], Eric Muller[1], Kirstin Dreise[1], Joyce Mgombelo[1] , and Ana Isabel Sacristán[2]

[1]Brock University, Department of Mathematics, St.Catharines, Canada; cbuteau@brocku.ca;
[2]Cinvestav, Mexico; asacrist@cinvestav.mx

This paper focuses on the process of university mathematics students engaging in a sequence of programming-based mathematical project tasks as part of a course. Data of this naturalistic research was collected mainly through four student projects and semi-structured individual interviews. The analysis led to narratives of students' development process (instrumental genesis) in which enacted strategies (instrumented actions) are highlighted. In this paper we discuss a participant's development process. Results suggest that the student, after 1 course, has appropriated programming as instrument for creating a tool for pragmatic purposes; however, not yet as instrument for mathematics investigations and applications, i.e., as an object-to-think-with (Papert, 1980).

Keywords: Programming, Instrumental Genesis, Strategies, University Mathematics Education.

Introduction

There is a resurgence of interest in integrating computer programming—more broadly, computational thinking (CT)—in education (e.g. in UK; Benton et al., 2017). This interest reflects how scientific fields, including mathematics, have developed computational counterparts (Weintrop et al., 2016), as well as the rise and need for proficiency in computational practices as 21st century skills. We see a crucial need to understand how students can be empowered to participate in such computational thinking, now an integral part of the mathematics and broader community.

In our work, we want to better understand how students come to appropriate programming as an instrument for mathematics investigations and applications "as mathematicians do". This appropriation involves students developing fluency (e.g. enacting strategies to progress forward) as they are (in the process of) engaging in this kind of mathematical work. This includes, not only becoming skilled at programming, but also designing and interpreting the mathematical work facilitated, or made possible, by programming. In this proposal, we discuss the case of a student's fluency development, analysed from the first-year data that was collected as part of a five-year study that examines how postsecondary mathematics students learn to use programming as a CT instrument for mathematics. It is a naturalistic (i.e., not design-based) research that takes place in a sequence of three project-based mathematics courses (called 'MICA') implemented in the mathematics department at Brock University (Canada) since 2001, where undergraduate mathematics majors and future mathematics teachers learn to design, program (e.g. in VB.net), and use interactive computer environments to investigate mathematics conjectures, concepts, theorems, or real-world applications (Buteau, Muller, & Ralph, 2015; Muller, Buteau, Ralph, & Mgombelo, 2009).

Conceptual Framework

Our work is framed by various interrelated concepts reflected in the literature on CT, CT in mathematics, and in mathematics education (Buteau, Muller, Mgombelo, & Sacristán, 2018). Wing (2014) defines CT 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" (p. 5), while Hoyles and Noss (2015) consider CT as abstraction, algorithmic thinking, decomposition, and pattern recognition. CT is an underlying process to computer programming described e.g. by Weintrop et al. (2016) as understanding, modifying, and writing codes (e.g., in *Python* or *C++*). Brennan and Resnick (2012) identify key dimensions for developing CT, namely *computational concepts* (e.g. iteration), *computational practices* (e.g. debugging projects or remixing others' work), and *computational perspectives* (e.g. a designer's evolving views about what could be programed).

CT has changed the nature of contemporary research in mathematics; for example, we now see computer-based proofs and new domains of research related to mathematics and computation such as bioinformatics. The European Mathematical Society (2011) recognizes this 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 modelling, simulation, optimization and visualization" (p. 2).

Based on their literature review and interviews with experts who use CT, Weintrop et al. (2016) outline what they believe to be the integral CT practices for mathematics and science, in particular encompassing this third pillar (Broley, Buteau, & Muller, 2017), across four main categories throughout which programming is an underlying practice: *data practices, modelling and simulation practices, systems thinking practices, and computational problem-solving practices*. E.g., the latter practice involves interpreting and preparing problems for mathematical modeling, assessing different approaches, developing modular solutions, and creating computational abstractions.

In the field of mathematics education, CT is not new; indeed, it has a 45-year legacy that started with the LOGO programming language (Papert, 1980) and extended into the theory of constructionism. Studies of constructionism in undergraduate-level mathematics education show how programming supports students' understanding of mathematical concepts and contributes to the development of critical thinking skills (e.g., Wilensky, 1995). As for students' processes when engaging in using programming for mathematical investigations and applications, we proposed a model based on insightful reflections on MICA student experiences (Buteau & Muller, 2010) which was later refined through a literature review (Marshall & Buteau, 2014); see Figure 1.

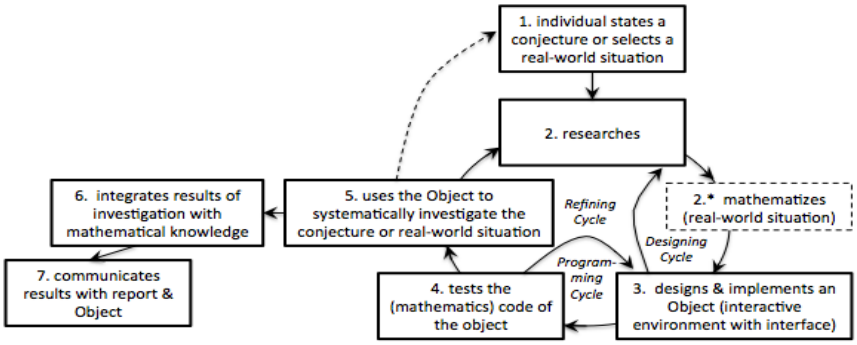

Figure 1. Development process model of a student engaging in programming for a mathematical investigation or application (Marshall & Buteau, 2014)

We use Lave and Wenger's (1991) concept of "legitimate peripheral participation," which describes how learners enter into a community of practice and gradually take up its practices, to understand how undergraduate students learn mathematics through CT activities. Based on this idea, mathematics is not knowledge to be acquired but rather is a process of participation through which the student

gradually gains membership to a community (of mathematicians). We thus 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), i.e., in the four CT practice categories. Thus, our work is focussed on how students (newcomers) engage in computational thinking for mathematics as mathematicians (elders) do. In fact, the proposed development process of a student engaging in programming for mathematical work as summarized in Figure 1 (removed of its numbering) seems to align with the process by which pure mathematicians use programming in their research work (Broley, Caron & St-Aubin, 2018).

In this proposal, we focus on this process and the strategies that students enact when engaging in CT-based mathematics work. In the instrumental approach framework (Rabardel, 1995/2002), the process is related to a student's *instrumental genesis* (Artigue, 2002): the process by which the student transforms an artifact into an instrument through schemes of usage and action. The instrumental genesis is a twofold process: instrumentalization –directed towards the artefact–, and instrumentation –directed towards the user–concerning "the development and appropriation of schemes of instrumented action which progressively take shape as techniques that permit an effective response to given tasks" (Artigue, 2002, p. 250). These schemes of instrumented action consist of technical and conceptual components. According to Drijvers, Godino, Font, and Trouche (2013), techniques may be considered "as the observable part of the students' work on solving a given type of tasks (i.e., a set of organized gestures)" (p.27), e.g., strategies used by a student at a certain step/cycle (see Figure 1) when engaging in a given programming-based mathematical task; and the schemes "as the cognitive foundations of these techniques that are not directly observable, but can be inferred from the regularities and patterns in students' activities"(p. 27). As Bozkurt et al. (2018) summarize:

> An artefact is initially not meaningful to the user until he or she develops associated schemes of instrumented action to use the artefact for achieving a task, and effectively turning the artefact into a useful mathematical instrument. (p.44)

E.g., turning (VB.net) programming into an instrument for mathematics investigation or application as mathematicians do, which we call 'a CT-instrument for mathematics'. In this proposal we discuss first results commencing to address our research question: *How do post-secondary students come to appropriate programming as a CT-instrument for mathematics?*

Methodology

Our research uses iterative design methods whereby some parts (participant recruitment and data collection) were designed in a way that would be least intrusive to (or constrained by) the natural learning environment. The study follows students' development over the course of (and beyond) their MICA I-II-III courses as they engage in 14 *exploratory object* (EO) mathematics project tasks, each resulting in an interactive environment and a report of mathematical findings (Muller et al., 2009). This paper focusses on a first analysis of data collected from the first year of our research. It thus draws from MICA I students where six (among 46) participants were recruited (voluntarily). In the MICA I course, there are 4 EO project tasks (which count for 71 % of students' final grades): 3 assigned individual ones, and a fourth one where students select the topic and can work in pairs or individually. Two-hour weekly labs progressively integrate through guided exercises the learning of computational concepts (variables, loops, etc.) in a mathematical context, whereas two-weekly hour

lectures introduce the math background needed for engaging in the EO tasks (Buteau et al., 2015).

Data from the participants included each participant's 4 project assignments (EO and report) and individual semi-structured interviews after completion of each of these EO tasks. The design of the interview guiding questions was informed largely by the students' development process model (Figure 1). In addition, data collected included weekly post-laboratory session online reflections (answering guiding questions) and an initial baseline online questionnaire before the beginning of MICA I course. Lab session and EO assignment guidelines were also collected and were complemented by an informal understanding of the MICA I course from the research team members' experiences in different capacity (researcher, instructor, course approver, former student).

All of the collected student data were analysed through thematic analysis techniques. Codes were developed based on categories informed by the conceptual framework (and associated literature). Each participant's interview and lab reflection data was coded individually by two coders, followed by a thematic analysis done jointly by the two coders. Themes were consolidated among the six participants' analyses, and led to sixteen themes regrouped in five main meta-themes, one of which concerned strategies. Narratives were composed, following the model steps from Figure 1, of each participant's process of engaging in their four EO projects, highlighting their enacted strategies, and overall summary. We now present preliminary findings by illustrating one of the six participants' development process with a focus on her use and development of instrumented techniques, i.e., *enacted strategies* (in italic, below), as a way to gain insights into her instrumental genesis towards appropriating VB.net programming into a CT instrument for mathematics.

Findings: the case of Hannah

Hannah (pseudonym) is a mathematics and computer science co-major, and thus had programming experience (in Java) prior to her MICA I course. In her first assigned EO project –exploring a conjecture of their choice about prime numbers or hailstone sequences–, Hannah struggled with the initial step of conjecturing (steps 1-2 in Figure 1); once she was passed this, she incrementally designed and programmed (steps 2-3) with ease her interactive environment (EO). When the time came to interpret the mathematics output of her program (steps 5-7), Hannah struggled and provided only data examples without mathematical explanation. In the EO 2 project on a RSA encryption application, Hannah reported feeling more satisfied and confident, finding it overall easier, in particular since she could start right away with designing and programming (i.e., step 1 was provided by the assignment guidelines). However again, her focus on the mathematics (step 5-7) seems to have been limited. We elaborate next on Hannah's engagement in her third EO task.

Hannah's development process and enacted strategies in her third assigned EO project task

Students are then asked 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 (Buteau et al., 2015). It follows an introduction of discrete dynamical systems in lectures, guided computer lab activities to program an EO to explore the logistic function system (quadratic, 1 parameter), including an instructor guided exploration of the system as the parameter is changed. At this time, students have been introduced to all basic computational concepts for mathematical work: variables, interface design, loops, conditionals, events, sequencing, and graphing. This assignment has a significant component on mathematical investigation. For Hannah, it is a critical learning opportunity since she

struggled with the mathematical components of the previous two EO projects.

Similarly to EO 2, the assignment guidelines provided the focus of the EO project (step 1 in Figure 1). Hannah voices that she did not research the related mathematics (step 2) when beginning the assignment because the lectures and labs were sufficient; however, in reflection, she believes she would have needed more time spent on the math concept to grasp it. In fact, Hannah missed some lectures due to illness and felt unable to catch up with the mathematical concepts. As in the past assignments, Hannah understands the programming concepts and is able to start designing and programming her project (step 3) in lab time by *remixing from the two previous labs* on the logistic function dynamical system. She used the *computational practice of incremental steps* with testing to complete the programming (programming cycle) – see Figure 2 for a screenshot of her EO:

> Hannah: the first step was setting up everything. … we have to setup the graph... regardless of … 'a' and 'b'… we're converting it to pixels instead of x and y's… So, we had to make sure that was working first before you could plug in anything.

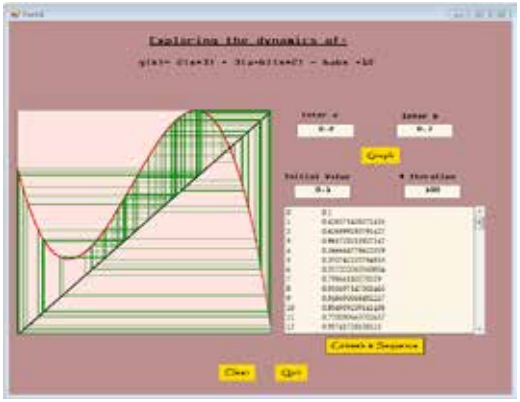

Figure 2. Screenshot of Hannah's third EO project about the 2-parameter cubic dynamical system

Hannah *tested the mathematics* output of her program *against other students' work and known examples from class* (step 4); e.g., Hannah says:

> Hannah: I used the, um, same values that, um, [the instructor] put in the labs. And, um, I checked, I compared the sequence and saw the math was working and then the graph and how it looks.

However, she had difficulty in investigating the given math topic or understand its meaning (steps 5 & 6), although she did *investigate* the cubic function *by systematically varying its parameters*:

> Hannah: I left 'b' fixed at zero and I increased 'a' a little bit and little bit and I see how it affected the curve….So, yeah, I played around with [a and b] until, um, until I got the fixed points.

but struggled with understanding the cobweb:

> Hannah: one of the challenges was, um, the, um, trying to figure out if it actually does converge. Because, um, the way we visualize it is, um, for each of the points. So, if you have 10 points, uh, 10 sequence numbers that you have, you draw lines between them. But, because we don't have like, sort of like, the starting point of the line and the ending point, you don't know if it is going that direction, which direction are going.

In the final part of the assignment, the report, Hannah's lack of understanding becomes clearer. Although her program works, her report does not reflect understanding; in particular, she *uses the*

program to generate an example, rather than elaborating calculations and drawing it by hand as asked (similar to how she completed her EO 2). Her report is lacking in explanations or conclusions to communicate her mathematical knowledge (step 7) and the discussion portion of the report is missing. This EO project could have been a great positive learning experience for Hannah to use programming for mathematics (learning) as this assignment was heavier on the mathematics investigation than previous ones. However, Hannah missed the opportunity partly due to external factors (illness).

Hannah's overall trend over the 4 EOs

Throughout the course, Hannah's focus seems to remain on programming rather than moving on to programming for mathematics, as revealed through her general approach to the 4 EO assignments. Hannah finds difficulty in identifying on her own, a mathematics topic relevant to a programming-based approach (step 1). She prefers to be given a conjecture (as in EO 2) and avoids creating her own when possible (as in EO 4). Once the mathematics problem or conjecture is sorted out, Hannah becomes much more comfortable. Most of the time, she researches thoroughly and uses resources to be well prepared for a solution before starting to program. She wants to understand the mathematics *a priori* and then program it, rather than using programming to understand (her general approach to 'interpreting and preparing problems for mathematical modeling' practice). This method causes difficulty in EO3 as programming for investigation is an intended way for students to understand.

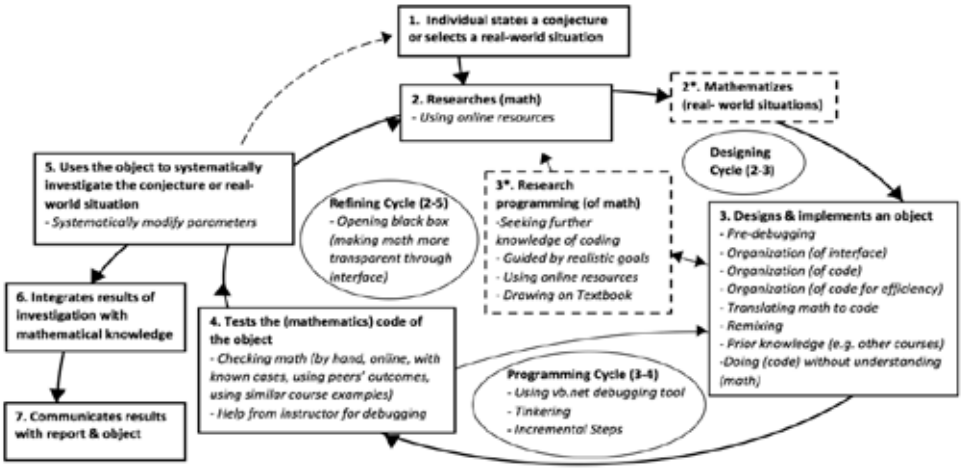

Figure 3. Hannah's summarized enacted strategies during the development of her 4 EO projects

Hannah uses her previous programming experience to help her complete the design and programming (step 3) with ease and little help from others: she *works incrementally* and *uses functions and modules to structure her program* (her approach to 'developing modular solutions' practice). Hannah developed multiple methods of *checking her mathematical work* including *using online resources*, *calculating by hand*, and *comparing known examples*. Because of her heavy preparation (in step 2) and strong programming skills (in step 3) there are not often many bugs to solve (step 4). When attempting to use the program to investigate the math problem (step 5) beyond the creation of the project, Hannah seems to lack confidence. She often isn't sure how or what to conclude from the program output. She doesn't seem to know how to use the program for investigation and wants to understand first, then "explore" later to confirm what she knows rather than to discover new ideas. Her limited investigations make it hard to integrate the results with her math knowledge (step 6). She finds some results noteworthy, but not necessarily what is intended as part of the investigation and

her insights, rather than mathematical, are more often related to programming. Lastly, when communicating her results in her written report (step 7), her descriptions are often short and not thorough in explanations. She provides examples that are either too simple to demonstrate knowledge or uses her program to generate an answer without further explanation of the process. Figure 3 summarizes Hannah's enacted strategies over the course of her 4 EO tasks.

Overall, Hannah seems to treat the course as a programming class for designing mathematics calculators rather than a mathematics class using programming as a tool for understanding and investigation, i.e. as an "object-to-think-with" (Papert, 1980). This perspective seems to limit her ability to dive in and explore new mathematical concepts deeply. In the final interview, Hannah said she intends and is looking forward to take the MICA II course. It will be interesting to see how (and if) she adapts her approach, enacting new strategies, in this more mathematically demanding course.

Conclusion

We have described Hannah's development process (including her strategies) as she engages with designing, programming, and using mathematics EOs. We observed that Hannah seems to appropriate programming as instrument for creating a tool for pragmatic purposes –i.e. as a calculator– however, not yet as instrument for mathematics investigations and applications –i.e., as an object-to-think-with (Papert, 1980). The findings provide insights on Hannah's development of fluency and instrumental genesis –i.e. how far she has come, through MICA I course, to appropriate (VB.net) programming as a CT instrument for mathematics investigations and applications. The transition from an instructionist to a constructionist approach to learning mathematics seems to have remained a challenge for Hannah, even with her previous programming background. In the near future and with the current trend, it is expected that students will arrive in mathematics department with programming skills. The case of Hannah can thus be considered as of a typical 'student' of the near future, and as such, provides insights of how these typical students' instrumental genesis may develop.

The first year student data analysis led to refine further the development process model (see step 2 and step 3* in Figure 3) to better capture the different strategies enacted by students. By highlighting how Hannah's strategies are enacted in the development process model, our study provides a beginning of how we can explore the role of strategies, as techniques, in students' instrumental genesis, leading next to study the related instrumented action schemes. The next steps, in our 5-year research, are also to bring together all of the thematic analysis findings and the development narratives for a deeper insight into students', including Hannah's, instrumental genesis, and to continue follow our participants into their upper-year MICA courses. It will also include the examination of instructors' adoption of learning environment (i.e., instrumental orchestration).

Acknowledgement

This work is funded by SSHRC (#435-2017-0367) and received ethics clearance (REB #17-088).

References

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.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and

mathematics: Some findings of design research in England. *DEME, 3*(2), 115–138.

Bozkurt, G., Uygan, C., & Melih T. (2018). Instrumental genesis of a preservice mathematics teacher: instrumented actions for perpendicular line construction in a dynamic geometry environment. *Proceedings of the 5th ERME Topic Conference MEDA 2018*

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* (1–25), Vancouver, Canada.

Broley, L., Buteau, C., & Muller, E. (2017, February). (Legitimate peripheral) computational thinking in mathematics. *Proceedings of the CERME* (pp. 2515-23), Dublin, Ireland.

Broley, L., Caron, F., & Saint-Aubin, Y. (2018). Levels of Programming in Mathematical Research and University Mathematics Education. *International Journal of Research in Undergraduate Mathematics Education*, *4*(1), 38–55.

Buteau, C. & E. Muller (2010): *Student Development Process of Designing and Implementing Exploratory and Learning Objects.* In *Proceedings of the Sixth CERME*, 1111–1120.

Buteau, C., Muller, E., Mgombelo, J., & Sacristán, A. (2018). Computational thinking in university mathematics education: A theoretical framework. *Proceedings of RUME,* San Diego, CA.

Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate math program at Brock University. In *Proceedings of Math+Coding Symposium*, London, ON.

Drijvers, P., Godino, J. D., Font, V., & Trouche, L. (2013). One episode, two lenses. *Educational Studies in Mathematics, 82*(1), 23–49.

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

Hoyles, C., & Noss, R. (2015). *Revisiting programming to enhance mathematics learning*. Paper presented at the Math + Coding Symposium, Western University.

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

Marshall, N. & C. Buteau (2014). Learning by designing and experimenting with interactive, dynamic mathematics exploratory objects. *Int'l J. for Technology in Mathematics Education*, 21 (2),49-64.

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.

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

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

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. (2014, January 10). Computational thinking benefits society. *Social Issues in Computing* [Web blog post]. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.