The 14th International Congress on Mathematical Education
Shanghai, 12th –19th July, 2020

RESEARCHING THE TEACHING AND LEARNING OF PROGRAMMING FOR UNIVERSITY MATHEMATICAL INVESTIGATION PROJECTS

Chantal Buteau, Brock University, Canada
Eric Muller, Brock University, Canada
Ghislaine Gueudet, University of Brest, France
Joyce Mgombelo, Brock University, Canada
Ana I. Sacristán, Cinvestav, Mexico

Since 2001, mathematics students and future teachers at Brock University (Canada) learn in a sequence of three courses to use programming for pure and applied mathematical investigations. In this presentation, we briefly highlight our research on the learning and teaching in these courses, and discuss how students are guided to progressively learn (i.e., develop a scheme) to articulate a mathematics process in a programming language.

SUSTAINED TEACHING OF PROGRAMMING FOR MATHEMATICS INVESTIGATIONS

For almost 20 years now, mathematics undergraduates and future mathematics teachers at Brock University (Canada) learn to design and program (e.g. in vb.net) interactive environments in order to use them for investigating a mathematics conjecture, concept, theorem or real-world application (Muller, Buteau, Ralph, & Mgombelo, 2009). As part of their *Mathematics Integrated with Computers and Applications* (MICA) I-II-III courses, students engage in completing 14 projects, 11 of which concerning assigned topics and investigations completed individually, and 3 of which about a topic selected by students and done in pairs (Buteau, Muller, & Ralph, 2015): e.g. the exploration of the dynamical system based on a 2-parameter cubic, the simulation of battles using the Lanchester equations, or cellular automata simulations of epidemics, and related costs – see (PROGMATICS research n.d.) for examples of MICA I-II-III assignment guidelines and original student final projects.

MICA courses have a format of 2 hours of lectures and 2 hours of computer labs, weekly. In lecture, MICA instructors introduce the mathematical content serving as background and motivation for students to develop, in labs and project assignments (often done on the student's own time), their programming-based mathematical investigations. This approach aligns with Papert's (1980) vision:

> "the relationship of teacher to learner is very different: the teacher introduces the learner to the [programmable math environment] in which discoveries will be made, rather than to the discovery itself." (Papert 1980b: 209)

The use of programming in the three MICA courses is, we argue, fully 'integrated'; in the sense that: i) MICA I students concurrently learn programming as they engage in mathematics problems requiring new programming concepts; ii) each of the programming-based mathematics projects takes its meaning, or its 'raison d'être' (Laborde, 2001), from programming technology; iii) the mathematics content was put together, from scratch, by faculty who use programming technology in their own mathematics research, with the aim of providing students with experiences similar to theirs; and iv) about 75% of students' final grades are based on their programming-based mathematics investigation projects.

The MICA course sequence offers a distinctively rich environment in which to research how students learn over time to use programming for mathematical investigations, which we elaborate next.

RESEARCHING TEACHING & LEARNING OF CODING FOR MATH INVESTIGATIONS

Our naturalistic research study (2017-2022) focuses on the following questions: i) How do post-secondary students come to appropriate programming as an instrument for mathematical investigations and applications? ii) Is their appropriation sustained over time (i.e. post MICA courses); if so, how?; and iii) how do instructors create a learning environment to support students' appropriation?

Our research is grounded on the instrumental approach theory (Rabardel, 1995). This approach frames students' learning processes with a main focus on the student's transformation of an artefact (in our case, programming) into a meaningful instrument (e.g., programming for mathematical investigations), which is thereby *appropriated* by the student. This process of appropriation involves students developing so-called 'instrumented-action schemes', which roughly can be described as a student's strategies and beliefs governing his/her strategies, for a given programming and mathematics goal. More precisely, following the work of Vergnaud (1998), a scheme is a stable organization of the subject's activity for a given goal. It comprises four components: i) the goal of the activity, sub-goals and expectations; ii) rules-of-action, generating the behavior according to the features of the situation; iii) operational invariants: concepts-in-action, which are concepts considered as relevant, and theorems-in-action, which are propositions considered as true; and iv) possibilities of inferences.

There are two main parts in a student's activity when using programming for mathematical investigation projects: the first requires creating a mathematics environment (e.g. selection and formulation of topic; design of the investigation; and coding of the environment, including coding of the mathematics; etc.), and the second using the environment to conduct the mathematics investigation (e.g. systematically changing parameters to observe behavior; interpreting the numerical/graphical output; explaining the results of the investigation; etc.). For students to appropriate programming as an instrument for mathematical investigations requires that they develop schemes for all of the steps involved in both of these parts (Buteau et al. 2019).

One of the steps involved in creating the mathematics environment part is to code a mathematical process that is, for the learner, an involving activity as stressed by Noss and Hoyles (1996):

> "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).

LEARNING TO CODE MATHEMATICS PROCESSES IN MICA I-II-III COURSES

We briefly discuss how MICA students are guided to progressively develop a scheme of articulating a math process in a programming language. We use data from our research for illustration purposes.

Guided initial learning of coding mathematics processes

Firstly in MICA I labs, students are asked to directly work on coding mathematics processes by initially being given a complete code and as the course progresses, the details of the given codes are reduced. In the lab guidelines, one can read: "do these [calculations] by hand *before* coding…", a guidance from the instructor seemingly for students to reflect on their own by-hand solution in order to then translate a mathematics process into code. We interpret this instructor guidance for students to develop their *scheme of articulating in vb.net a mathematical process* (goal), with a strategy of starting by 'translating' in vb.net what s/he would do by hand (rule-of-action), that is governed by the belief that 'a programming language can work in a similar manner as one works by hand' (theorem-in-action) (Buteau et al. 2019).

For example, in lab 3, students learn about loops and conditional controls, and are given a complete code for reproducing (and fixing a minor bug) about checking the primality of an integer. Jim, a first-year student in MICA I course, says:

> [I] do believe that, assuming I knew how to use the Mod command, I would have (given time) been able to make something resembling it from scratch, as the logic of how it searches for primes has already been touched upon in class. (Jim.Lab3)

By lab 5, students are provided with partial step-wise pseudocodes for calculating powers in Z_n. In lab 6, students are provided with incomplete pseudo-codes for calculating the gcd of two integers and Euler's phi function, and then no codes at all for determining the inverse of elements in Z_n. Jim says:

> In terms of difficulty, I wouldn't so much describe the lab as difficult as I would call it long. (Jim.Lab6)

suggesting that he successfully coded all of the requested mathematical processes without difficulty (note that Jim was ill and completed his lab work on his own). Overall, Jim's excerpts suggest, that he developed a scheme to articulate in vb.net a mathematical process, with components similar to what was intended (i.e. as described above).

Further developing codes of mathematics processes for assigned mathematical investigations

Secondly, beyond the labs in MICA I course, students are exposed (from week 4 on) to programming-based mathematical investigation projects requiring them to code at least one new mathematics process, sometimes by combining and adapting (i.e., remixing) codes developed in previous labs. In other words, students further learn to code mathematics processes as they are engaged in a larger aim, i.e. a mathematical investigation involving many mathematical processes. E.g. the first project builds on lab 3: state or select a conjecture about primes, and create a program in vb.net to investigate it. Jim describes about his investigation of the Pólya conjecture concerning odd prime factors of integers:

> I would start with the system to check prime factors and I would build on that system to check every number, then build on that and so on. (JIM.Project1.#18)

Jim, once more, seems to use his strategy of starting by 'translating' in vb.net what he would do by hand.

In MICA II-III courses, students are expected to code more complex mathematics processes as part of investigation projects – i.e., there is no longer exercises about coding mathematics processes during lab sessions, rather only work on projects that require coding mathematics processes. For example, students are guided to code the Buffon problem concerning the probability that a needle dropped on a slatted floor crosses a crack, and are then required to modify it to a square-tiled floor (i.e. Buffon-Laplace problem). Another example is the coding of battle simulation (a linear system of differential equations) modified to include random troops sent to the battle (and examine how it impacts the results of the battle). Kassie, a MICA II student, mentioned about the latter:

> I took like the [Lanchester] equations … and tried to think about it like, in a programming kind of mindset… [The random troops] was a little bit more difficult… I didn't understand and then I did, so I kind of just took that... I did the code the way I thought of it. (Kassie.Project8.#7,27)

Coding mathematics processes as part of original mathematical investigation projects

Thirdly, every MICA course ends with an original project on a topic selected by the student or pair of students. This means that students are then coding related mathematics processes from scratch, which may be overwhelming for some students depending on their selected topic. In that part of the course, the instructor plays a role of mentor to each student or pair of students, sometimes helping with this coding but (also) importantly with the mathematization or constraining of the topic for a programming-based mathematical investigation manageable by the students at that stage.

CONCLUDING REMARK

We hypothesize that this mentoring role by the instructor during the final projects may be key to students' appropriating programming as an instrument for authentic mathematical investigations. That is for students to come to program mathematics not as an end in itself but rather as a means to engage in mathematics. Our ongoing research aims in part at shedding light on this.

ACKNOWLEDGMENTS

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

References

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. (2019). University Students Turning Computer Programming into an Instrument for 'Authentic' Mathematical Work. *International Journal of Mathematical Education in Science and Technology.* DOI: 10.1080/0020739X.2019.1648892

Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate math program at Brock University. Retrieved at http://researchideas.ca/coding/docs/ButeauMullerRalph-Coding+MathProceedings-FINAL.pdf

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.

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: Kluwer.

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

PROGMATICS research (n.d.). Teaching resources for university mathematicians. www.ctuniversitymath.ca/category/teaching-resources/

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

Vergnaud, G. (1998). Toward a cognitive theory of practice. In A. Sierpinska & J. Kilpatrick (Eds.), Mathematics education as a research domain: A search for identity (pp. 227–241). Dordrecht: Kluwer Academic.