Students using programming for pure and applied mathematics investigations

Chantal Buteau, Laura Broley, Kirstin Dreise, Eric Muller

▶ **To cite this version:**

Chantal Buteau, Laura Broley, Kirstin Dreise, Eric Muller. Students using programming for pure and applied mathematics investigations. Épijournal de Didactique et Epistémologie des Mathématiques pour l'Enseignement Supérieur, 2023, 2 | 2023, 10.46298/epidemes-9190 . hal-03601988v3

HAL Id: hal-03601988
https://hal.science/hal-03601988v3

Submitted on 14 Feb 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ÉpiDEMES – 2023, 2 submitted 08/27/2021, accepted 06/02/2022, revised 07/02/2023

Students using programming for pure and applied mathematics investigations

Chantal Buteau, Laura Broley, Kirstin Dreise, & Eric Muller,

Abstract. In this paper, we recount our research on undergraduate mathematics students learning to use programming for mathematics investigation projects. More precisely, we focus on how a particular theoretical perspective (the Instrumental Approach) helps us better understand this student activity. Pulling data from students' and instructors' experiences in a sequence of courses (offered since 2001), our results expose, at the micro and macro levels, how the student activity is organized (through stable 'ways of doing'), and highlights the complexity of this activity (as an intertwined web of 'ways of doing' involving a combination of both mathematics and programming competencies). We end with concrete recommendations to instructors.

Keywords. Programming, Mathematics Investigation, Project-Based Learning, Learning Process, Instrumental Approach

Résumé. Dans cet article, nous présentons notre recherche sur les étudiants de premier cycle en mathématiques apprenant à utiliser la programmation pour des projets d'investigation en mathématiques. Plus précisément, nous nous concentrons sur la façon dont une certaine perspective théorique (l'Approche instrumentale) nous aide à mieux comprendre cette activité de l'étudiant. S'appuyant sur des données des expériences d'étudiants et d'instructeurs dans une séquence de cours (offerts depuis 2001), nos résultats décrivent comment l'activité de l'étudiant est organisée (par le biais de «façons de faire» stables), et met en évidence la complexité de cette activité (comme un réseau entrelacé de « manières de faire » impliquant une combinaison de compétences en mathématiques et en programmation). Nous terminons par quelques recommandations concrètes pour les instructeurs.

Mots-clés. Programmation, Investigation mathématique, Apprentissage par projet, Processus d'apprentissage, Approche instrumentale

Contents

1. Jim does “it”

Prime numbers are an interesting mathematical topic to many mathematicians. While mathematics is full of patterns and predictability, prime numbers appear to be an exception: Mathematicians have long pondered what patterns they follow and when the next prime number will appear, among other questions.

Jim was interested in understanding the nature of prime numbers. To this end, he decided to study the Pólya conjecture, which claims that for any integer, *n*, at least 50% of the natural numbers less than *n* will have an odd number of prime factors. Through research online, Jim knew that the conjecture had been disproven and that the smallest known counterexample is 906,150,257. Since it is impractical to study such numbers by hand, Jim used his programming knowledge to create a computer program to do the required calculations (of how many prime factors a number has and how many numbers less than n have an odd number of prime factors). Soon, Jim had a program that would output whether the Pólya conjecture was true for each number *n* in a user inputted range. He was able to use his program for small positive integer input; however, the program lagged and crashed for larger inputs. Jim found that his desired level of efficiency required more programming knowledge than he had, limiting his current research to relatively small numbers. Commenting about how strange it is that a pattern should suddenly breakdown after 900 million cases, Jim still found value in using his program to explore and reflect on the nature of prime numbers.

Now consider: Who do you think Jim is? Perhaps a mathematician, or a professor? A master’s or 4th-year university student working on a thesis? These are common positions in which we might expect to find someone completing such an investigation.

It turns out that Jim is a 1st-year undergraduate student who conducted this investigation as part of a course workload. This leads us to wonder: What knowledge or skills does a student, such as Jim, develop or need to engage in this kind of activity? What kind of learning environment could promote such engagement? And how does it compare to the work of a research mathematician?

2. Professional mathematicians certainly do "it" too!

When Jim chose to study the Pólya conjecture, by designing and coding[1] a program to explore it, and then reporting his findings, he was walking in the footsteps of some research mathematicians who select a problem of interest in pure or applied mathematics and investigate it using software they create. This approach to mathematical research was recognized in 2011 by the European Mathematical Society: "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). Later, in 2016, mathematicians participating in a 6-month 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)

In that same year, Weintrop et al. (2016) proposed a categorization of the computational practices used by professional mathematicians and scientists (see Figure 1), 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).

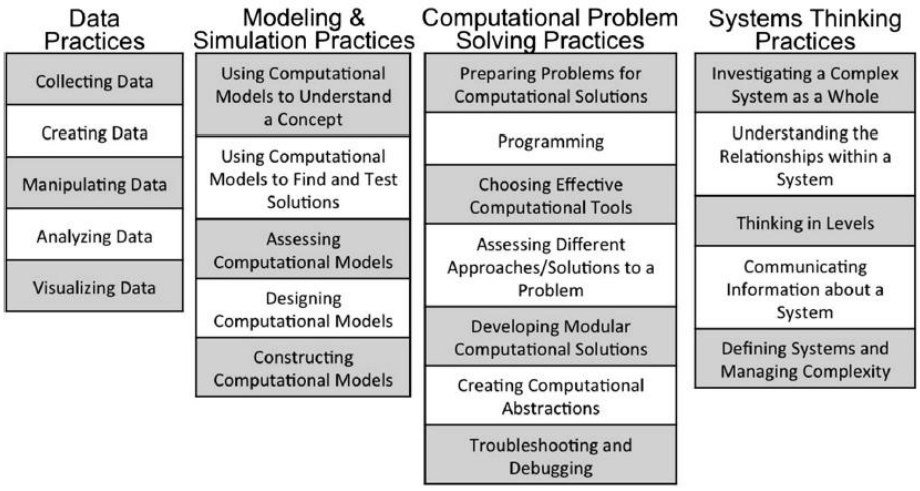

Figure 1 – Taxonomy of computational practices in mathematics and science (Weintrop et al., 2016, p. 135)

[1] In the literature 'programming' and 'coding' are often used interchangeably. We see a distinction between the two (although it is sometimes difficult to identify): programming involves the design of a program and its 'coding' in a certain programming language (cf: Armoni, 2016).

To get a glimpse of the range of mathematics encompassed by Weintrop and al.'s (2016) taxonomy, Broley et al. (2017) selected examples of four mathematicians whose research could be categorized according to each of the four taxonomy groupings. Under *Data Practices*, the mathematician used extensive historical data of companies listed on stock exchanges to produce probability outcomes for portfolios. Another applied mathematician used *Modeling & Simulation Practices* to study muscles when she designed "a system of equations that would allow her to study the features of interest, i.e. tensions, bulges, and fibres; but only once the model was implemented on the computer" (Broley et al., 2017, p. 2518). In pure mathematics, a researcher was engaged in *Computational Problem Solving Practices* to generate from within all permutations of a given length *n* those that do not contain a permutation of specified numbers *x* (of length less than *n*). And, as a final illustration, a mathematician researching celestial mechanics employed *Systems Thinking Practices* as he studied the three-body problem with perturbations.

We can identify the use of some of these computational practices in Jim's study of the Pólya conjecture. In *Data Practices*, Jim created the data needed to explore the conjecture; in *Modeling & Simulation Practices*, he used a computational model to understand the concept of the conjecture; and in *Computational Problem Solving Practices*, he used most of the practices, at his level of programming (he would not have used any of the *Systems Thinking Practices*). In other words, it seems that Jim's engagement in using programming in mathematics shares features (in terms of practices) with the engagement of professional mathematicians (Broley et al., 2017; Weintrop et al., 2016).

In reporting the results of a Canada-wide survey, Buteau et al. (2014) pointed to the possibility that research mathematicians use programming much more in their research than they do in their teaching. Nevertheless, some university mathematics departments are offering courses that engage students in programming for their mathematical activities.

3. Some university mathematics programs have integrated "it"

The place of programming in university mathematics programs can vary from one university to another. Programming may or may not be a requirement for all mathematics students – e.g., through a compulsory computer science course – and it may be integrated in different ways – e.g., in certain mathematics courses or throughout an entire program; as a tool or as an object of study (Modeste, 2015). For instance, a 2016 survey of 46 mathematics departments in the U.K. found that 89% of undergraduate mathematics programs teach programming to all students and 78% of the mathematics courses required for an undergraduate science degree had some programming-based component, most commonly in numerical analysis or statistics (Sangwin & O'Toole, 2017). In comparison, the University of Oslo (Norway) recently has revised its mathematics programs so that programming is used as a transversal problem-solving tool in courses throughout the programs (Malthe-Sørenssen et al., 2015).

This kind of integration in which programming skills can be developed and used across courses throughout a university mathematics program was implemented at Brock University (Canada) in 2001. Inspired in part by the increasingly computational nature of research mathematicians' work (as described above), Brock's Department of Mathematics and Statistics developed an innovative concentration called *Mathematics Integrated with Computers and*

Applications (MICA; Ralph, 2001). This includes three core project-based courses—MICA I, II, and III—in which students create their own computer programs to engage in pure or applied mathematics investigations (Buteau, Muller, & Ralph, 2015). See the Appendix for details about the distribution of 14 projects across the three MICA courses and examples of the topics covered, among which 11 individual projects on assigned topics and 3 original end-of-course projects done individually or in pair on a topic of their own choice. Jim's investigation of the Pólya conjecture was initiated in the first of these 14 projects.

In the MICA courses, students' completion of the programming-based mathematics investigation projects is carefully scaffolded. The projects themselves are designed to progressively build in complexity, in terms of the mathematics and the programming. Moreover, each week, students attend 2-hour lectures (where the professor introduces the mathematics and programming concepts required for the projects) and 2-hour labs in which students may engage in additional activities for learning programming concepts, and where they can work on the projects amongst peers, teaching assistants, and/or professors (Buteau & Muller, 2014; Buteau, Muller, & Ralph, 2015).

This careful scaffolding has been a particularly important issue because most students begin their university mathematics programs with little to no prior experience in programming, let alone in using programming for mathematical activities. But a recent international trend in school curricular reform suggests that this may not be an issue in the coming years.

4. School curricula are getting in the game: Universities, prepare for "it"!

In her closing keynote at the *Mathematics Education in the Digital Age* (MEDA) conference in 2018, Celia Hoyles called for all university mathematics departments to prepare for a new student generation: in the coming years, each new cohort of first-year students will have more computer programming skills than the cohort before, specifically in mathematics. This call comes from the fact that elementary and secondary schools around the world are increasingly integrating programming—or more broadly, computational thinking (cf: Wing, 2006)—into their curricula.

At the elementary level, programming recently has been introduced into various curricula (Dagienė et al., 2019), both on its own and as a part of mathematics (Misfeldt et al., 2020). For example, programming is now integrated into the algebra strand of the mathematics curriculum in Ontario, Canada (Ontario Ministry of Education, 2020) and the geometry strand of the mathematics curriculum in France (Vandevelde & Flückiger, 2020). In comparison, in England (Benton et al., 2017) and in New Zealand (New Zealand Ministry of Education, 2020), programming is integrated separately from mathematics as part of a distinct computing or technology curriculum. Meanwhile, Finland uses a cross-curricular approach in which programming is explicitly taught in mathematics and crafts, and it is also expected to be applied as a tool across all subjects (Bocconi et al., 2018). Most of these curricula begin to introduce programming at age 6 (or Grade 1).

The students experiencing these new elementary curricula will be well prepared for secondary computer science courses, which have been available for much longer. Italy, for example, has a long history of teaching computer science and integrating it into compulsory mathematics courses at the secondary level, beginning in the 1980s (Forcheri et al., 1990). In many countries, computer

science courses traditionally have been offered as separate elective courses, but there is a recent shift towards integrating programming in mathematics and making it compulsory. France has integrated "algorithmics" (including the study of algorithms, programming, and some other elements of computer science) into high school mathematics courses since 2009, specifically for students aiming for a STEM path in postsecondary education (Lagrange & Rogalski, 2017). Poland does not include programming in mathematics but recently has integrated it through computer science courses in compulsory secondary programs (Webb et al., 2017). In the Canadian context, the province of British Columbia now offers Grade 11 and 12 computer science (with mathematics) courses as elective mathematics credits that can count towards students' graduation requirements in mathematics (British Columbia Ministry of Education, 2020); and, most recently, the Ontario Ministry of Education reformed its Grade 9 mathematics curriculum to include programming within the algebra strand, required for all students (Ontario Ministry of Education, 2021).

These are only a few examples of how programming is being integrated at the school level. By 2016, many other countries had also completed or were planning to complete a curricular reform to include computational thinking (Bocconi et al., 2016). Figure 2 provides a depiction of this, in and around Europe.

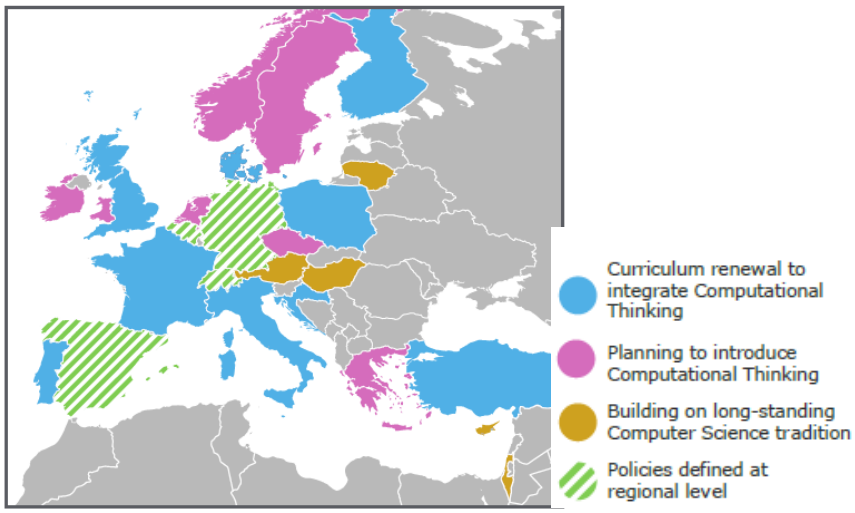

Figure 2 – Integration of computational thinking in and around Europe (Bocconi et al., 2016, p. 27)

The dramatic uptake of computational thinking in school curricula around the world is driven by different arguments (Passey, 2017), many of which are grounded on the belief that all students need preparation for the ever-increasing presence of digital technologies in 21st century life.

Interestingly, integrating programming as part of mathematics in particular, is supported by long-standing research dating back to the work of Papert (1980) with Logo, a programming language designed for children, to turn the computer into an "object-to-think-with". This includes research showing that programming for mathematics at the university level may support students' learning in many areas such as algebra (Leron & Dubinsky, 1995), calculus (Clements, 2020), probability (Wilensky, 1995), statistics (Mascaró et al., 2016), and combinatorics (Lockwood & De Chenne, 2020). More generally, Noss and Hoyles (1996) have stressed that any learner, when engaging in modifying a program, articulates relationships between concepts involved in the program, "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).

Considering the context sketched above—from the computational nature of some mathematicians' research, through the increased integration of programming in schools, and, finally, to the affordances of programming for university mathematics learning—it is important to think about how university mathematics students might (be supported to) learn to use programming for their mathematical activities. This is the focus of our work.

5. What are we, the authors, doing about "it"?

We are mathematicians in a mathematics department in a Canadian university who have been involved in the development (Muller), teaching (Buteau), and early studies (Muller and Buteau) of the MICA courses mentioned previously. We are also mathematics didacticians with diverse expertise, such as the instrumental approach (Gueudet and Santacruz-Rodriguez), university mathematics education (Gueudet and Broley), programming in mathematics education (Sacristán and Broley), constructionism (Sacristán), and mathematics cognition and learning (Mgombelo). Over the years, we have included many research assistants (e.g., Dreise), graduate students, and a post-doctoral fellow (Broley), some of whom were previous MICA students (e.g., Dreise and Broley). Despite our diverse expertise and backgrounds, we have a common interest in examining the learning and teaching in university mathematics education like the MICA courses.

Since about 2017, our team has been conducting research addressing the following questions:

1. How do students come to appropriate programming as an instrument for authentic pure or applied mathematical investigations?
2. Is this appropriation sustained over time (i.e., past course requirements)?
3. How do instructors create a learning environment that supports students' appropriation of programming as an instrument for mathematical investigations?

Note that these research questions are directly linked to the initial wonderings we posed in relation to Jim's investigation of the Pólya conjecture, concerning the knowledge or skills a student would develop or need to engage in such an investigation, as well as the kind of learning environment that would promote it. These questions also use the term 'instrument' with a particular meaning that will be explained later.

In this paper, we discuss results that are related to the first research question and that were recently presented at the International Network for Didactic Research in University Mathematics (INDRUM) 2020 conference (Buteau et al., 2020; Gueudet et al., 2020). At the conference, we were interested in investigating a more precise sub-question:

> What do we learn about the activity of students using programming in an authentic mathematical investigation by using the theoretical frame of the instrumental approach, considering programming as an artefact?

Before being able to discuss results related to such a question, we first need to clarify some of the elements contained in the question. What exactly do we mean by "the activity of students using programming in an authentic mathematical investigation"? What is this "theoretical frame of the instrumental approach" and what is an "artefact"? We answer these two questions in the next two sections.

6. This is "it"!

As a first-year student in MICA I, Jim does "it": He uses programming for various mathematics investigations. As mentioned previously, some professional mathematicians certainly do it too. We take the position that these experts' work process provides a model for how students can engage in the use of programming for mathematics investigation (Broley et al., 2017; Weintrop et al., 2016). This is what we mean by "authentic" investigation. Seen from the perspective of literature on "authentic learning," Jim's engagement encompassed a majority of the recognized themes, as summarized by Rule (2006):

> real-world problems that engage learners in the work of professionals; inquiry activities that practice thinking skills and metacognition; discourse among a community of learners; and student empowerment through choice. (p. 1)

To further understand Jim's learning through authentic investigation, we use a development process model (dp-model; Figure 3) designed by Buteau and Muller (2010) to describe university students' engagement in programming-based mathematics investigation tasks. It was initially developed from an analysis of tasks given to students in the MICA courses and has since been refined through a literature review (Marshall & Buteau, 2014) and an analysis of student data (Buteau et al., 2019). In the model, the term "object" (e.g., see step 3) is used to refer to an "object-to-think-with" (Papert, 1980). In our case, the object is an interactive program (designed and coded by a student) with the goal of mathematical investigation. We often refer to it as an "exploratory object" (EO).

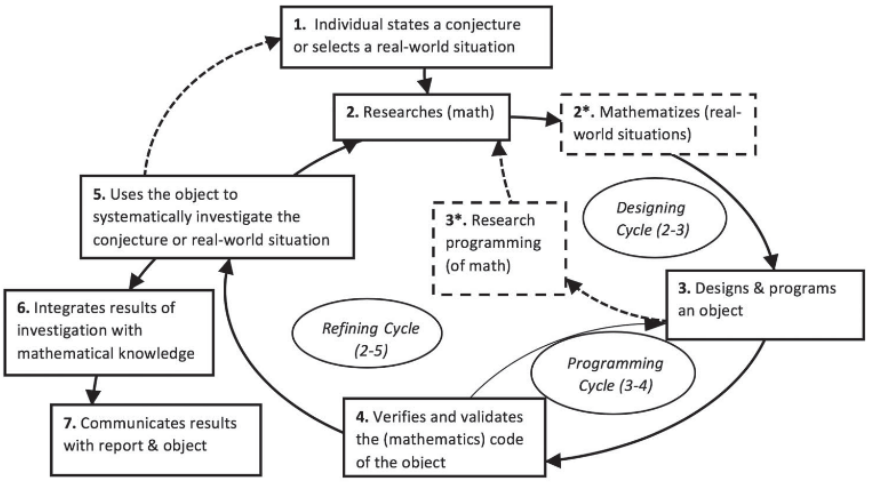

Figure 3 – Development process (dp-)model of a student engaging in programming for a pure or applied mathematics investigation (Buteau et al., 2019, p. 1025)

There is a short (less than 5-minute) video presenting a dynamic illustration of the dp-model in the context of particular student projects (see Balt & Buteau, 2020b). As an additional example, we describe what Jim did for his third MICA I project (as inferred through interviews with him and his final project report). In the third MICA I project, the topic (exploring a cubic in a dynamical system) was given to Jim through project guidelines (step 1). Jim did not need to do additional

research on the mathematics topic because, as he said, "the labs as well as the lecture notes were very sufficient for this" (A3.2[2]; step 2). To design his EO, Jim used classroom resources, including his lecture notes and responses to his questions during labs (A3.13). He also remixed a program that was created in an assigned lab activity. He explained: "I basically used that lab as a template and then just built off of that" (A3.4). He further commented that it was not too difficult to "program it in steps and just chunk, chunk it all together and make the program" (A3.4; step 3). Jim practised incremental testing and debugging as the need arose throughout his programming. He shared insight about this method, noting that "of course with computer programming you kind of accept that you are going to run into some bugs at some point, which is why you know you test it at different stages" (A3.15; programming cycle 3-4). In this project, Jim used a type of testing (step 4) that he hadn't used in previous projects, in which he compared his program to online technology:

> I had tried using other graphic methods, mainly an online graphing calculator called Desmos, and tried graphing something similar or related to the function just to get an idea, um, and then put it in the actual program and, you know, saw that they were very similar. (A3.24)

Once his program was completed and functional, Jim used two strategies to investigate the mathematics. He used a guess and check method, and also made gradual changes based on his observations (A3.33; step 5). Jim's report is thorough and contains his insights with connections to his understanding of the mathematics. For example, when describing the two functions involved in the program, he said: "The graphs of $f(x)$ and $g(x)$ share an uncanny resemblance to one another, however considering that $f(x)$ is meant to be $g(x)$ over a range of $[0,1]$, this should come as no surprise" (EO3.9; steps 6 & 7).

The general tasks in ***pure mathematics research*** *involving programming:*

1. choice and delimitation of a mathematical problem;
2. creation of a computer program;
   a) development of an algorithm;
   b) coding of the algorithm;
   c) verification and validation of the program;
3. exploration;
   a) observation of mathematical objects;
   b) formulation of conjectures;
   c) verification of conjectures;
4. proof;
5. communication of the results and preparation for the future.

The general tasks in ***applied mathematics research*** *involving programming:*

1. choice of a real phenomenon;
2. development of a model;
   a) study of the phenomenon and existing models;
   b) identification of important aspects;
   c) analysis of real data;
   d) creation of a mathematical model;
   e) analytical validation of the model;
   f) calculation of parameters;
3. programming of the model;
   a) research on existing computer tools;
   b) development of an algorithm;
   c) coding of the algorithm;
   d) verification and validation of the program;
4. search for solutions and experimentation with the model;
5. interpretation and validation of results;
6. communication of the results and preparation for the future.

[2] This reference indicates the place where the cited information is found in our research data. "A" ("EO") indicates that the information is from an individual interview (report) concerning a particular project. The number after "A" ("EO") specifies the project (out of 14) and the number after the point specifies the segment of the interview (report). "L" indicates that the information is from a lab reflection. The number after "L" specifies the reflection (out of the total number of reflections).

Figure 4 – Models of the research process where programming is used as a tool in pure (left) and applied (right) mathematics (adapted from Broley, 2015, pp. 37, 56; translated from French); the tasks are not necessarily encountered in an equal, linear, or distinct manner in each research project

We claim that as Jim engaged in the steps of the dp-model, he was working similarly to a professional mathematician: If one uses programming in one's own pure or applied mathematics research, the dp-model is likely not surprising; indeed, it aligns well with the collection of practices identified by Broley (2015) based on interviews with 14 Canadian mathematicians on how they use programming in their research work (see Figure 4).

While the description using the dp-model gives us insight into what Jim did on a general level (i.e., his activity), we are interested in digging deeper to understand how and why Jim did it this way. To explore this, we combine the dp-model with the tools offered by the instrumental approach (Rabardel, 1995, 2002) and the theory of conceptual fields (Vergnaud, 2009), which we describe next.

7. How we look at "it"

In short, the instrumental approach (Rabardel, 1995, 2002), provides a lens to analyse how a student learns while using a given technology. Key to this lens is a distinction between an *artefact* (a simple object, like a calculator) and an *instrument* (an artefact that has become a purposeful tool). In mathematics education, the first studies that used the instrumental approach examined the learning processes involved when secondary school students transformed a graphing calculator into an instrument (Guin et al., 2005). These studies showed how when faced with a specific mathematical goal (e.g., determine the limit at infinity of a function), students learned how to use the artefact (e.g., plot the graph of the function, adjust the window for large values of x, etc.), and they learned mathematics at the same time (e.g., a graph with a horizontal asymptote indicates that the function has a limit at infinity). In our work, the artefact is a programming language (e.g., python or vb.net), together with an integrated development environment (e.g., PyCharm or Visual Studio), which facilitates the use of the language (e.g., through compiling programs, providing debugging tools, offering features for creating user interfaces, etc.). We are interested in identifying how a student like Jim transforms this artefact into an instrument for authentic mathematics investigations.

In the instrumental approach, this transformation of interest is referred to as *instrumental genesis* and involves the development of so-called (*instrumented-action) schemes*. In our context, schemes can roughly be described as a student's ways of doing (i.e., their strategies and the beliefs governing their strategies) for a given programming and/or mathematics goal. More formally speaking, schemes are defined as stable organizations of a subject's activity for a given goal (Vergnaud, 1998, 2009). They are described through four essential components:

1. the *goal* of the activity, sub-goals, and anticipations;
2. *rules-of-action*, generating the behaviour according to the features of the goal (roughly speaking, the student's strategies and actions);
3. operational invariants, governing the rules-of-action: *concepts-in-action*, which are concepts considered as relevant, and *theorems-in-action*, which are propositions considered as true

(roughly speaking, the student's beliefs and understandings that explain or govern what they do); and

4. possibilities of *inferences*—that is what permits the adaptation of the scheme to the precise features of the situation.

Schemes provide a theoretical model of how a subject learns "in activity", i.e., while engaging in a task. For a given subject (e.g., a student), a class of situations is defined by the general goal of their activity (e.g., draw the graph of a function). As the subject interacts with artefacts (e.g., a graphing calculator) to reach this goal in a specific situation (e.g., draw the graph of the function $f(x)=sin(x)$), they may develop a scheme (of use of the calculator), which they may later mobilize in a similar situation through inferences. Sometimes the adaptation of the scheme to a new situation can lead to the emergence of new operational invariants, new rules-of-action, or even a new scheme. Identifying these different components of schemes is what allows us to dig deeper into the "how" and the "why" behind a student's use of programming in authentic mathematics investigations.

It is important to note that the notion of scheme comes from Vergnaud's theory of conceptual fields, which is a more general theory of students' learning of mathematics. In our work, we refer also to this more general theory, identifying not only schemes of use of a programming language, considered as an artefact, but also what we might call "mathematical schemes" (i.e., schemes for accomplishing mathematical goals, without the use of an artefact). In our case, the general goal of the students' activity is "to investigate a complex situation, combining mathematical knowledge and programming." By using programming for this general goal, we claim that students develop an instrument, associating some aspects of programming and schemes of use for specific sub-goals associated to different steps of the dp-model (Buteau et al., 2019).

But how can we possibly capture this? More specifically: How can we gain access to the schemes developed and used by a student (like Jim) in an authentic mathematics investigation? In the next section, we outline the methodology we adopted to achieve our goal.

8. How we are researching "it"

Our 6-year long research examines student and instructor activities in a natural environment (i.e., not designed for a research purpose). It follows students' development over the course of (and beyond) their MICA I–II–III courses (described previously) as they engage in mathematics investigation projects (14 in total). It also examines MICA instructors' teaching. To date, our research has included an in-depth study (described below) of individual students' progression through each of the MICA courses, and a broader study of MICA students' perceptions and experiences obtained through questionnaires administered at the beginning and at the end of each MICA course. Our approach was designed to be least intrusive to (or constrained by) the natural learning environment provided by the MICA courses, such as for participant recruitment and data collection. This means in particular that we do not intervene in student task design (investigation projects), nor during lectures or labs.

In this paper, we focus on the case of Jim, a student enrolled in the MICA I course, from the first year of our research. In that first year, six among 46 MICA I students voluntarily participated.

We collected data including each participant's four projects (i.e., their computer programs and written reports), and individual semi-structured interviews after the completion of each project. For the interviews, we used guiding questions incorporating "explicitation" techniques (Vermersch, 2006) to help students relive their actions during the development of their investigation project. These questions aligned with the dp-model (Figure 3). For example, the interviews contained the question: "How did you know that your program worked?" in alignment with step 4 of the dp-model. Student participants were also asked to write reflections (prompted by guiding questions) after each weekly lab session, fill out an initial baseline questionnaire, and participate in an additional interview before the beginning of MICA I. In addition, we collected all course material.

Our naturalistic methodological approach has the benefit of studying activity that occurs in a natural institutional context (with all of its constraints). It also has a limitation: we do not directly observe participants' project engagement. We mitigate this limitation by triangulating all the data available on each participant (Gueudet et al., 2020).

To begin our scheme analysis, we started with Jim's data because he was particularly reflective in his responses. We note that Jim was not exceptional in terms of achievement (both in mathematics and in programming). We first analyzed his early data (baseline questionnaire; baseline interview; lab 1–4 reflections; EO1 interview; EO1 project) for an initial identification and description of schemes (this was done individually by two parties, and then consolidated): we did so by trying to observe, in Jim's declarations during his interviews, elements of schemes: goals; descriptions of how he acted to achieve those goals (rules-of-action); and reasons for acting this way (operational invariants). We compiled the schemes and their components in a table, and then re-ordered the table according to the dp-model. The whole data (for all six participants from MICA I) was also analysed using thematic analysis techniques. It was coded (individually by two coders) using codes developed from our broader theoretical framework (Buteau et al., 2019) or emerging from the data. Codes were then regrouped in themes (jointly by the two coders). After consolidating themes among the six participants' analyses, we had 16 sub-themes regrouped in five main themes; two of these corresponded to strategies and perceptions. Using codes from the strategies theme (associated to rules-of-action) and the perceptions theme (associated to operational invariants), the initial table of Jim's schemes was refined and chronologically extended to his whole MICA I data (i.e., to include his other 3 projects). This analysis of Jim's schemes also included the programming of his projects (i.e., the computer programs and the written reports).

After all that, what did we find out about how Jim engaged in using programming for the authentic mathematics investigations in MICA I? In the next section, we employ the theoretical tools introduced above and the results of our study to provide some illustration.

9. How Jim does "it"

Jim was a first-year student in the Bachelor of Science Honours in Mathematics program at Brock University. From his baseline questionnaire, we noted that his outlook towards mathematics and technology was very positive prior to entering the MICA I course. Formally, Jim had very limited knowledge of computer programming, except for some exposure in his youth with Logo programming. Through the MICA I course, Jim completed all four assigned mathematics investigation projects (described in the Appendix: projects 1-4 in MICA I). His MICA I original

final project (fourth project) concerned the investigation of the Buffon needle problem about the probability that a needle randomly dropped on a floor with parallel slats would cross over two slats. In the following, we focus on Jim's engagement in his first MICA project, in which he investigated the Pólya conjecture.

The first 4 weeks of the MICA I course prepared Jim and his peers for their first project. In lectures, Jim was mainly exposed to prime numbers and hailstone sequences, and to conjecturing about these concepts. In lab sessions, he started learning about the basics of programming in vb.net: variables, loops, conditional statements, and create, read from, and print in a graphical user interface (GUI). Starting in lab 3, Jim and his peers were progressively guided to code mathematics; for example, in lab 3, they were asked to reproduce and fix the code for checking the primality of an integer, and in lab 5, they were given a partial pseudo-code for powers in Z_n. The first project directly built on lab 3 and asked Jim and his peers to state or select a conjecture about primes, and then to create a program in vb.net to investigate it.

After attempting to formulate a conjecture of his own, Jim decided to investigate the Pólya conjecture (steps 1–2 in Figure 3). He then designed a numerical experiment and programmed it in vb.net (step 3); his code included a series of nested loops and conditionals to check how many prime factors a number has and how many numbers less than n have an odd number of prime factors, and concluded with conditionals to output whether the conjecture was true for each number, n, in a user inputted range. Jim created his initial program incrementally by coding one part, then testing it to make sure it worked correctly (step 4), then moving on to the next part. He then attempted to refine the program to create better organization and to increase efficiency, as his program lagged and crashed when large numbers were inputted. However, he found that his desired level of efficiency required more coding knowledge than he had, limiting his research to small numbers (step 5). Nevertheless, Jim still found value in using his program as it still allowed him to explore and reflect on the nature of prime numbers (step 6). In his report (step 7), Jim exclaimed "how strange [prime numbers] can be to follow a pattern for the first 900 million examples, only to abandon it thereafter" (EO1.5).

Now if we zoom in on different steps, we can find examples of different schemes. We will use the following components of a scheme as described in the section "How we look at 'it'": goals, rules-of-action, concepts-in-action, and theorems-in-actions. We present three "particularly well-selected" examples (yes, you'll understand in the next section why we picked those ones!).

9.A. Example 1

Let us first zoom in on Jim's step 1 above. Jim's initial work involved attempting to *formulate a mathematical conjecture (goal)*. To accomplish this goal, Jim started by trying to understand better the concept of primes through a thorough investigation:

> At first, I was trying to kind of think of trying to understand more about the nature of primes before I would really do my conjecture. (A1.2)

Jim's investigation included searching on the internet and taking notes. Jim's persistence in his investigation suggests that he believed that to formulate a conjecture, a good understanding of the concepts involved was necessary. We interpret Jim's activity as a rule-of-action, governed by a

theorem-in-action, as shown in Table 1. It is possible to consider the goal "investigate a mathematical concept" as a sub-goal, for Jim, of the goal "to formulate a mathematical conjecture" (we just zoomed in further!). This sub-goal is associated with a sub-scheme, with rules-of-action like "search on the internet." We do not present here this sub-scheme.

After investigating the mathematical concept, Jim tried to represent prime numbers and to observe a pattern:

> I had some other notes pertaining to me trying to figure out this conjecture basically where I would plot out the primes and look for any patterns of how they worked. (A1.3)

We interpret Jim's mathematical activity again as rules-of-action, probably developed through working on many problems in prior mathematics experiences. It seemed that Jim believed that an appropriate representation of prime numbers would help him to find a pattern. We interpret this belief as a theorem-in-action. The concepts of "representation" and "pattern" are relevant for Jim in this situation and guide his activity: they can be considered as concepts-in-action (which are explicit here).

We also note that Jim's overall approach to formulating a mathematical conjecture comprises using investigation and searching to learn more about prime numbers. This suggests a general belief that he could learn mathematics through exploration. We include this as another theorem-in-action.

We summarize the components of Jim's scheme in Table 1.

Rules-of-action	Investigate the mathematical concept (search on the internet, take notes); search for a representation; search for a pattern
Concepts-in-action	Representation; pattern
Theorems-in-action	Understanding related concepts helps to formulate a conjecture; an appropriate representation is helpful to find a pattern; I can learn mathematics by exploration

Table 1 – Jim's scheme for formulating a mathematical conjecture (Gueudet et al., 2020)

9.B. Example 2

Next, let us zoom in on Jim's step 3 above. As part of designing and implementing his object, Jim needed to *articulate a nested process in a programming language (goal)*. Jim and his peers already had been introduced to the articulation, in vb.net, of a process involving repetitions using loops. Jim suggested that after this, he had to build on this introduction to develop an approach to articulating, in vb.net, a nested process:

> We actually went over how to build this kind of system [involving repetitions] in class. So, the only thing new about the project was kind of learning how to nest them, properly structure them, to make this running program. (A1.18)

According to him, Jim codes nested loops to articulate a nested process, which suggests a belief that a nested system can be processed by programming technology as nested loops. In addition, Jim seemed to indicate coding such nested loops incrementally because incremental coding helps to properly structure the nested loops. We interpret Jim's activity as two rules-of-action, along with

their governing theorems-in-action (see Table 2). In this situation, we identify "nested system," "nested loops," and "loops" as explicit concepts-in-action in Jim's activity. For example, when reflecting on his experience, Jim emphasized the importance of understanding

> this idea of nested kind of system and how to build upon a single system into multiple ones. … Like one system inside another and I think that's pretty key. … It [is] one of those inherent things. (A1.27)

This scheme seems to be, for Jim, at the core of programming. This suggests Jim's awareness of mobilizing or developing it further in his future programming-based mathematics investigations.

Rules-of-action	Code nested loops articulating the nested process; code them incrementally
Concepts-in-action	Nested system; nested loops; loop
Theorems-in-action	A nested system can be processed by programming technology as nested loops; incremental coding helps to properly structure the nested loops

Table 2 – Jim's scheme for articulating a nested process in a programming language (Gueudet et al., 2020)

9.C. Example 3

Finally, we zoom in on Jim's step 3 again, but we focus on what Jim did to *articulate a mathematical process in the programming language (goal)* as he was developing a program to explore the Pólya conjecture. Having researched the mathematics required for this exploration, Jim described how he approached the programming of the mathematics in vb.net:

> I basically tried to organize and sort out what needed to be programmed but I kind of realized as I was going, I kind of knew everything that needed to be done. It just required a set of system nested within each other so once I know that I had to figure out how to program each individual system. This one check[s] for prime. This one is a loop … that sort of thing. (A1.8)

Jim seems to indicate that he organized this mathematical process (described in Section 1) as a nested system, which required him to understand the concept of "mathematics and programming as nested systems" and was guided by the belief that a mathematical process can sometimes be seen as a nested system. Jim then decomposed the nested system into individual processes before beginning to write the code for each of them one at a time. We suggest this activity was governed by Jim's belief that to program a nested mathematical process, one can break it down and individually code the smaller parts. Jim also indicated a belief that programming and mathematics as systems have embedded layers. For example, Jim explains

> how sometimes you will have a variable within a sine function … or a nested loop, kind of one layer down. It is like something within a greater system. (A1.11)

Jim's decomposition and programming were also supported by his understanding of the concepts of "decomposition of a system" and "individual processes."

As mentioned above, in lab 3, prior to beginning their first project, Jim and his peers were asked to re-type the code of checking the primality of integers and to fix a minor bug planted in the

code. This was their first encounter of a coded mathematical process in the course. In reflection, Jim said:

> [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. (L3)

We interpreted this as indicating that Jim would start by "translating" in vb.net how he would do it by hand, which seems to demonstrate Jim's belief that a programming language can work in a similar manner as one works by hand. These were identified as a potential rule-of-action and theorem-in-action. Jim then put them in action when he was completing his first project. He described his programming by enumerating different processes that seem to align with a by-hand method:

> 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. (A1.18)

We synthesize the components of Jim's scheme in Table 3.

Rules-of-action	Organize the mathematical process as a nested system; decompose the nested system in individual processes before programming; code individual processes; start by "translating" in the programming language how I would do it by hand
Concepts-in-action	Mathematics and programming as nested systems; solving-by-hand method; decomposition of a system; individual process
Theorems-in-action	A mathematical process can be seen as a nested system (i.e., made of many parts); to program a nested mathematics process, one can break it down and individually code the smaller parts; a programming language can work in a similar manner as one works in mathematics by hand; programming and mathematics as systems have embedded layers

Table 3 – Jim's scheme of articulating a mathematical process in the programming language (Gueudet et al., 2020)

In the above three examples, we illustrate three schemes in the context of Jim's first project, but we also observed these schemes in Jim's later investigation projects. Sometimes the schemes were stable (i.e., we observed the same rules-in-action, theorems-in-action, and concepts-in-action), and other times the schemes evolved (i.e., we observed new or modified rules-in-action, theorems-in-action, or concepts-in-action), depending on the task that Jim faced.

Now let us remember our research question: What do we learn about the activity of students using programming in an authentic mathematical investigation by using the theoretical frame of the instrumental approach, considering programming as an artefact? We aim to understand students' engagement in using programming in an authentic mathematics investigation. Drawing on the three examples studied above, we discuss some elements of answering this question and addressing this aim in the next section.

10. What we found out about "it"

Studying Jim's activity using the theoretical frame of the instrumental approach illuminates the existence of schemes of different natures. To see what we mean, let us compare the goals of each example above:

- Example 1: *formulate a mathematical conjecture*
- Example 2: *articulate in a programming language a nested process*
- Example 3: *articulate a mathematical process in the programming language*

In Example 1, Jim's goal was strictly mathematical; in Example 2, his goal was strictly related to programming; in Example 3, his goal combined aspects of both programming and mathematics. It turns out that, along his entire engagement in authentic mathematics investigations using programming, all the goals we identified for Jim had one of these three natures. This led us to categorize the schemes and to name them: m-schemes, p-schemes, and (p+m)-schemes, accordingly.

Of particular interest to us are the (p+m)-schemes because they can be considered as bridging mathematics and programming skills. The identification of theorems-in-action and concepts-in-action corresponding to (p+m)-schemes in particular deepens our understanding of how mathematics and programming relate to each other in programming-based mathematics investigations. Gerianou and Janqvist (2019) argue that the theory of instrumental genesis, and schemes in particular, allow to bridge mathematical and digital competencies. Our study illustrates and confirms this, in the specific case of programming technology.

Another interesting thing we reflect on based on our observations is the complex dialectics between stability and evolution of the scheme components developed by students. We consider that the students in the MICA course (and Jim in particular) already had stable m-schemes; for example, they might have met before situations in mathematics classes where they would have needed to formulate a conjecture. These m-schemes would be adapted to the features of the new situation involving programming, but the organization of the activity would remain stable. In contrast, programming was a new activity for the MICA students; the p- and (p+m)-schemes we identified were most likely developed during the MICA course.

Nevertheless, we claim that some of the rules-of-action and operational invariants in p- and p+m-schemes are already stable at the end of the MICA I course. Indeed, we have observed their mobilization on several occasions, and we consider that this confirms their stability. For example, we found evidence that, in his second and third MICA I projects, Jim used his rules-of-action for the (p+m)-scheme described above, while also developing additional rules-of-action (e.g., 'I ignore coding special cases of the mathematics process that are not needed for the mathematical investigation') and theorems-in-action (e.g., 'Special cases in the mathematics code potentially leading to bugs, but that do not affect the mathematical investigation, can be ignored').

11. More things we found out about "it"

Recall that the schemes we identified are related to the different steps of the dp-model (Figure 3; Buteau et al. 2019). For instance,

- *formulate a mathematical conjecture* relates to step 1
- *articulate in a programming language a nested process* relates to step 3
- *articulate a mathematical process in the programming language* also relates to step 3

In addition to these three schemes, our case study of Jim also led us to identify 18 other schemes corresponding to all steps of the dp-model. We do not present all these schemes in detail here, for the sake of brevity. However, identifying all these schemes highlighted another important feature of students' engagement in using programming for mathematics investigations.

Studying Jim's overall engagement reveals that the different schemes developed by students along the MICA courses are interrelated; they constitute a complex structure. To capture the dynamics of this structure, we created a 5-minute video (Balt & Buteau, 2020a) that highlights how the dp-model (Figure 3) can be seen as the skeleton of the complex structure. It also highlights how schemes are connected in different ways.

Schemes can be connected "horizontally" in the sense that they are called upon within a very dynamic and non-linear process. For example, three schemes that could be involved in such a process are "articulating a mathematical process in the programming language," "validating the programmed mathematics," and "debugging." Indeed, when one programs a mathematical process, one would write a first code and run it to check that it is working correctly. This often doesn't happen in the first trial, and one would have to debug and fix the code (several times?) and go back to validate that the mathematics works.

Schemes also can be connected "vertically" in that they constitute different levels of generality. For example, "designing and programming an object" is a goal (step 3 of the dp-model), which is associated with a single scheme. The two schemes from Examples 2 and 3, namely "articulate in a programming language a nested process" and "articulate a mathematical process in the programming language," can then be seen as sub-schemes (associated with sub-goals). It is possible to consider more precise goals and obtain sub-schemes (like in this example); it is also possible to consider more general goals, which lead to "super-schemes." For example, in the case of Jim, we also identified a scheme labelled "to formulate a conjecture about primes," which is a sub-scheme of the Example 1 super-scheme presented above.

In sum, the activity of using programming for pure or applied mathematics investigation projects involves a web of schemes with what we've called "vertical" and "horizontal" connections as outlined above (and as dynamically illustrated in the video). The development of such a web of schemes by a student (like Jim) represents their instrumental genesis; that is, their transformation of programming into an instrument for pure and applied mathematics investigations. Using the instrumental approach has thus provided us with a deeper understanding of students' engagement in using programming for pure or applied mathematics investigations—not only by exposing, at the micro and macro levels, how this engagement is organized (through rules-of-actions) and why (through concepts- and theorems-in-action), but also by elaborating the dp-model as a composition of goals of different natures, through which a complex web of schemes is highlighted. As with any research process, there is much more to learn, and we expect the instrumental approach to continue to be helpful in this regard.

12. Words of wisdom to instructors of "it"

To end our paper, we would like to provide some ideas for university mathematics instructors who may wish to support students in using programming for various mathematics investigations. These ideas are inspired not only by our own experiences—as researchers, developers, instructors, and/or students of the MICA courses—but also by the insights of others (e.g., participants of our research and other scholars).

One key insight came from another part of our research (Buteau et al., 2021), where we shared our results with another developer and experienced MICA instructor, Bill Ralph. Given that Bill is also a research mathematician using programming in his research work, he implicitly knew about the schemes students may develop for using programming as an instrument for mathematics investigations. Making the schemes explicit, however, seemed to encourage some useful reflections on his instructional practices. For instance, when reflecting on our identified rules-of-actions for articulating a mathematical process in the programming language, Bill said:

> It's funny talking to you … I wish I had thought more about pushing them to the big picture. I have said things like that, but I haven't really really emphasized it … I wish I had said more and more about pencil and sitting with a sheet of paper for five seconds and thinking about how this is all gonna work. I think that would be helpful.

We expect that other instructors could also find it useful to think about their students' learning in terms of the development of schemes grounded in the dp-model. This could include instructors not only integrating a project-based learning approach, but also those delivering any mathematics course involving programming, such as computational data science courses. Similar to Bill, such instructors could be shown the proposed theoretical framework presented in this paper and the results to get started (e.g. through reading this paper).

Other insights that we have collected from over 20 years of sustained implementation and evolution of the MICA courses concern three specific features of a learning environment that is conducive to students' appropriation of programming as an instrument for authentic mathematics.

First, we have found that it is beneficial for the environment (e.g., the course(s)) to be developed around mathematics, rather than programming. When programming is the focus (e.g., in a compulsory computer science credit), it is likely that students will not learn to apply their skills to their mathematical activities (e.g., they will not develop the crucial (p+m)-schemes mentioned previously). Indeed, many of the research mathematicians in Broley's (2015) study recalled that they did not learn to use programming for mathematics until after their undergraduate degree; one mathematician, for example, said of his undergraduate experience that included a computer science course: "Even if I did a little programming, it was never well mixed with my mathematics" (p. 121; translated from French). Papert's (1996) original vision for using programming in mathematics education suggests that computing should be a source of power for students: "Starting with computing as a formal thing to understand and then come to the application later, takes away its power" (Barba, 2016, para. 25). The potential benefits of combining programming with mathematics have also been confirmed to us by other instructors and students. For example, a

research mathematician who recently had designed a course integrating programming-based explorations and mathematics concepts suggested that

> when [the students] put the two together then there's some kind of synergy that takes place where each one informs the other one. And it's a much more educational experience for them: they really learn a lot more. (Broley, 2015, p. 95)

In a similar vein, a MICA II student explained that the MICA courses

> teach you how to use an interactive programming environment … and allows you to use it to investigate different mathematical theorems and concepts. It is very effective because it allows you to make your own program to be able to see how this concept works, and play around with it to reach a further understanding of the concept. (Buteau, Muller, & Marshall, 2015, p. 147)

Second, we have found that programming-based mathematics investigations can be supported well by a flexible project-based curriculum, where students are invited to engage in authentic projects (reflecting the dp-model), developed based on the instructors' expertise and interests. In such an environment, the instructor still has the important role of providing support to students, in terms of ensuring projects are at an appropriate level for students' abilities (in programming and in mathematics), and in terms of offering helpful interventions as the students complete the projects (Hoyles & Noss, 1992). Nevertheless, as Papert (1980) points out, "the relationship of teacher to learner is very different: the teacher introduces the learner to the [programming environment] in which discoveries will be made, rather than to the discovery itself" (p. 209). In other words, the main role of the instructor is not to present ideas but rather to create conditions that promote invention, creativity, and the pursuit of ideas (Barabé & Proulx, 2017). Once again, the potential benefits that can come from doing it this way have been confirmed to us by instructors and students. For instance, the same mathematician mentioned above expressed his feeling that

> it's much better if the students can program it for themselves. If they're sitting in front of the screen and they can play with it and they can adjust parameters, it becomes a kind of a game and it's more interactive for them. And it's better than me just showing them a picture at the front of the classroom, you know what I mean? If they're doing it themselves, they learn it way better. (Broley, 2015, p. 90)

In reflecting on his entire MICA I-II-III experience, another MICA student explained in more detail how he was empowered by engaging in authentic tasks:

> In almost any scenario, the creation of a mathematical model on a computer program can be made to simulate, test, explore, or discover any dynamical system ... MICA has opened my learning pathway to explore the possibility of being able to create models for major companies to be used for research purposes. ... The possibilities are only limited to the creativity of the mathematician making the models. I think the major skill I will take with me from MICA courses is the ability to create, analyse, and explore dynamical systems and make the connections between them and the real world. (Buteau et al., 2016, p. 161)

Finally, we have found that supporting students' appropriation of programming as an instrument for authentic mathematics investigations requires adequate space in the curriculum. Researchers have long discussed the tension teachers experience between constraining students' engagement to ensure that intended course content is covered and supporting students' freedom of exploration in constructionist experiences (Noss & Hoyles, 1996). Research mathematicians confirm that they experience this tension at the university level (Broley et al., 2018). Our own experiences with the MICA courses tell us that to integrate programming-based mathematics investigation projects in a course, and to support students' development of the complex web of schemes outlined above, it is important to alleviate the tension. In particular, one should (a) avoid overloading the course curriculum (e.g., each MICA course focuses on four or five course projects); and (b) plan for significant classroom time dedicated to students engaging in authentic projects (e.g., the 2-hour weekly labs in the MICA courses).

Over 40 years ago, Seymour Papert (1980) presented what he called a "radical" vision, whereby computer programming would enable mathematics students to turn the computer into an "object-to-think-with," "to *do* something and figure things out, in a dance between the computer and [their] thoughts" (Barba, 2016, para. 25). Today, this vision is not so radical. At a time when computation provides a powerful tool for some professional mathematical work, school curricular reforms in the name of computational thinking abound, and we know that university mathematics students can be supported in learning to use programming to engage in mathematics investigations, we are encouraged to say: Let's do "it"!

Acknowledgements

This work is funded by SSHRC (#435-2017-0367) and has received ethics clearance (REB #17-088). We first thank all of the research assistants for their valuable work toward our research project. Our thanks also go to mathematician Bill Ralph (Brock University) who generously agreed to participate in the research and who regularly provides us with further insights, such as through reading and commenting on this manuscript. We also sincerely thank the three mathematicians who kindly agreed to read our manuscript and who provided us with rich comments: Hichem Ben-el-Mechaiekh (Brock University), Bernard Hodgson (Université Laval), and Yvan St-Aubin (Université de Montréal).

Appendix

The following table shows examples of the 14 projects completed by MICA students (adapted from Buteau, Muller, & Ralph, 2015, p. 2). As depicted, the projects may evolve over the years and according to instructors' mathematical interests.

MICA	Project	Topic
1	1	Conjecture about primes or hailstone sequence
	2	RSA encryption method
	3	Discrete dynamical system (cubic with two parameters)
	4	**Original, end-of-term project** with student-chosen topics with student-chosen topics: e.g., - Encryption pseudorandom-ness

		- The effects on the economy of central banks' monetary policy according to the Price Adjustment Model
Cohorts		***(Ralph's 2011-12 cohort)***	***(Fuks' 2014-15 cohort)***
2	5	Buffon needle problem & Monte Carlo integration	Discrete equations: Model of water pollution in a system of two connected lakes connected by a stream
	6	Stats application to stock market	Systems of discrete equations: models of the spread of infectious diseases
	7	Synchronization of traffic lights	Dynamical system of the logistic function & bifurcation diagram
	8	Markov chains applied to income demographics and chronic illness	Stochastic models of bacterial growth
	9	**Original, end-of-term project** with student-chosen topics: e.g., - Is it better to run or walk in the rain? - The Mandelbrot Set - Simulation of rowing-boat running time	
3	10	Dynamical system of the logistic function & bifurcation diagram	Cellular automata models of road traffic flow
	11	Simulation of battles (Lanchester equations)	Empirical modes and curve fitting: investigations of the Zipf's law.
	12	Prey-predator biological model (Lotka-Volterra)	ODE models in pharmacokinetics, Michaelis-Menten equation
	14	Cellular automata, simulation of epidemics & costs	Pursuit problems in 2D, Hathaway's circular pursuit
	15	**Original, end-of-term project** with student-chosen topics: e.g., Changes in the water supply of a local lake over time	

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. https://tinyurl.com/ttx5xamz

Armoni, M. (2016). Computer science, computational thinking, programming, coding: The anomalies of transitivity in K-12 computer science education. *ACM Inroads, 7*(4), 24–27. https://doi.org/10.1145/3011071

Balt, K., & Buteau, C. (2020a, September 4). *Illustrating the web of schemes: A student's process when engaging in using programming for pure or applied math investigation* [Video]. YouTube. https://youtu.be/teJAd3TDw9E

Balt, K., & Buteau, C. (2020b, September 4). *Using programming for pure/applied mathematics investigation: Mandelbrot set and running in the rain illustrations* [Video]. YouTube. https://youtu.be/irTlCE-eXhc

Barabé G., & Proulx J. (2017). Révolutionner l'enseignement des mathématiques: Le projet visionnaire de Seymour Papert [Revolutionizing mathematics education: Seymour Papert's

visionary project]. *For the Learning of Mathematics*, *37*(2), 25–30. https://flm-journal.org/Articles/318D3351495F8F94E785130E59D4CE.pdf

Barba, L. A. (2016, March 5). *Computational thinking: I do not think it means what you think it means*. Lorena A. Barba Group. https://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*, 115–138. https://doi.org/10.1007/s40751-017-0028-x

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice* (JRC Science for Policy Report). European Commission. https://doi.org/10.2791/792158

British Columbia Ministry of Education. (2020, July). *B.C. graduation program policy guide: Grades 10 to 12*. https://tinyurl.com/2kreee9h

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. https://papyrus.bib.umontreal.ca/xmlui/handle/1866/12574

Broley, L., Buteau, C., & Muller, E. (2017, February). (Legitimate peripheral) computational thinking in mathematics. *Proceedings of CERME 10*, 2515–2522. https://tinyurl.com/3d33wnc4

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), 33–55. https://doi.org/10.1007/s40753-017-0066-1

Buteau, C., Gueudet, G., Dreise, K., Muller, E., Mgombelo, J., & Sacristán, A. (2020, September). A student's complex structure of schemes development for authentic programming-based mathematical investigation projects. *Proceedings of INDRUM 2020*. https://hal.archives-ouvertes.fr/hal-03113837/document

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. https://doi.org/10.1080/0020739X.2019.1648892

Buteau, C., Jarvis, D., & Lavicza, Z. (2014). On the integration of computer algebra systems (CAS) by Canadian mathematicians: Results of a national survey. Canadian Journal of Science, *Mathematics, and Technology Education, 14*(1), 1–23.

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 CERME 6*, 1111–1120. http://ife.ens-lyon.fr/publications/edition-electronique/cerme6/wg7-07-buteau-muller.pdf

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* (pp. 163-185). Springer. https://doi.org/10.1007/978-94-007-4638-1_8

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

Buteau, C., Muller, E., Marshall, N., Sacristán, A., & Mgombelo, J. (2016). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Digital Experiences in Mathematics Education, 2*(2), 142–166. https://doi.org/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 the Math + Coding Symposium* [Symposium]. London, ON, Canada.

Buteau, C., Muller, E., Santacruz Rodriguez, M., Mgombelo, J., Sacristán, A., & Gueudet, G. (2021). *Instrumental orchestration of using programming for authentic mathematics investigation projects* [Manuscript in preparation]. Department of Mathematics and Statistics, Brock University.

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

Clements, E. (2020). *Investigating an approach to integrating computational thinking into an undergraduate calculus course* (Doctoral dissertation, University of Western Ontario). Scholarship@Western. https://ir.lib.uwo.ca/etd/7043

Dagienė, V., Jevsikova, T., & Stupurienė, G. (2019). Introducing informatics in primary education: Curriculum and teachers' perspectives. In S. Pozdnyakov & V. Dagiene (Eds.), *Informatics in schools: New ideas in school informatics* (pp. 83–94). Springer. https://doi.org/10.1007/978-3-030-33759-9_7

Drijvers, P., Godino, J. D., Font, V., & Trouche, L. (2013). One episode, two lenses. *Educational Studies in Mathematics, 82*(1), 23–49. https://doi.org/10.1007/s10649-012-9416-8

European Mathematical Society. (2011*). Position paper of the European Mathematical Society on the European Commission's contributions to European research*.

Forcheri, P., Furinghetti, F., & Molfino, M. T. (1990). Integration of computer science and mathematics in upper secondary school: reflections and realizations. *Computers & Education, 14*(4), 325−333. https://doi.org/10.1016/0360-1315(90)90044-8

Geraniou, E., & Jankvist, U. T. (2019). Towards a definition of "mathematical digital competency". *Educational Studies in Mathematics, 102*(1), 29–45. https://doi.org/10.1007/s10649-019-09893-8

Gueudet, G., Buteau, C., Muller, E., Mgombelo, J., & Sacristán, A. (2020, September). Programming as an artefact: What do we learn about university students' activity? *Proceedings of INDRUM 2020*. https://hal.archives-ouvertes.fr/hal-03113851/

Guin, D., Ruthven, K., & Trouche, L. (Eds.). (2005). *The didactical challenge of symbolic calculators: Turning a computational device into a mathematical instrument*. Springer. https://doi.org/10.1007/b101602

Hoyles, C. (2018, September 5). *Mathematics education in the digital age: Promise and reality* [Plenary talk]. Fifth ERME Topic Conference on Mathematics Education in the Digital Age, University of Copenhagen, Copenhagen, Denmark.

Hoyles, C., & Noss, R. (1992). A pedagogy for mathematical microworlds. *Educational Studies in Mathematics, 23*(1), 31–57. https://doi.org/10.1007/BF00302313

Lagrange, J. B., & Rogalski, J. (2017). Savoirs, concepts et situations dans les premiers apprentissages en programmation et en algorithmique [Knowledge, concepts and situations in early learning in programming and algorithmics]. *Annales de Didactiques et de Sciences Cognitives, 22*. https://hal.archives-ouvertes.fr/hal-01740442/document

Leron, U., & Dubinsky, E. (1995). An abstract algebra story. *American Mathematical Monthly, 102*(3), 227−242. https://doi.org/10.1080/00029890.1995.11990563

Lockwood, E., & De Chenne, A. (2020). Enriching students' combinatorial reasoning through the use of loops and conditional statements in python. *International Journal of Research in Undergraduate Mathematics Education, 6*,303–346. https://doi.org/10.1007/s40753-019-00108-2

Malthe-Sørenssen, A., Hjorth-Jensen, M., Langtangen, H. P., & Mørken, K. (2015). Integration of calculations in physics teaching. *Uniped, 38*, Article 6. https://tinyurl.com/ywx9vvsw

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.

Mascaró, M., Sacristán, A. I., & Rufino, M. (2014). Teaching and learning statistics and experimental analysis for environmental science students, through programming activities in R. In G. Futschek & C. Kynigos (Eds.), *Constructionism and creativity: Proceedings of the 3rd International Constructionism Conference* (pp. 407–416). Österreichische Computer Gesellschaft.

Misfeldt, M., Jankvist, U.T., Gerianou, E., & Bråting, K. (2020). Relations between mathematics and programming in school: juxtaposing three different cases. In A. Donevska-Todorova, E. Faggiano, J. Trgalová, Z. Lavicza, R. Weinhandl, A. Clark-Wilson & H.-G. Weigand (Eds.) *Proceedings of the MEDA2020 conference* (pp. 255-262), Linz University.

Modeste, S. (2015). Impact of informatics on mathematics and its teaching. On the importance of epistemological analysis to feed didactical research. In Gadducci, F., Tavosanis, M. (Eds.) *History and Philosophy of Computing, Series: IFIP Advances in Information and Communication Technology, Vol.487*, Springer).

New Zealand Ministry of Education. (2020). *Technology in the New Zealand curriculum (revised Technology learning area)*. https://tinyurl.com/k5fxcz4n

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Kluwer. https://doi.org/10.1007/978-94-009-1696-8

Ontario Ministry of Education. (2020). *Elementary curriculum: Mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Ontario Ministry of Education. (2021). *Mathematics, Grade 9*. https://tinyurl.com/3djym45h

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

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123. https://doi.org/10.1007/BF00191473

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies, 22*, 421–443. https://doi.org/10.1007/s10639-016-9475-z

Rabardel, P. (1995). *Les hommes et les technologies: Approche cognitive des instruments contemporains* [People and technology: A cognitive approach to contemporary instruments]. Armand Colin. https://hal.archives-ouvertes.fr/hal-01017462/document

Rabardel, P. (2002). *People and technology: A cognitive approach to contemporary instruments* (H. Wood, Trans.). Université Paris 8.

Ralph, W. (2001). Mathematics takes an exciting new direction with MICA program. *Brock Teaching, 1*(1), 1. https://tinyurl.com/3r4a5xbt

Rule, A. C. (2006). Editorial: The components of authentic learning. *Journal of Authentic Learning, 3*(1), 1–10. https://tinyurl.com/3vse7mub

Sangwin, C. J., & O'Toole, C. (2017). Computer programming in the UK undergraduate mathematics curriculum. *International Journal of Mathematical Education in Science and Technology, 48*(8), 1133–1152. https://doi.org/10.1080/0020739X.2017.1315186

Vandevelde, I., & Fluckiger, C. (2020). L'informatique prescrite à l'école primaire. Analyse de programmes, ouvrages d'enseignement et discours institutionnels [Computers prescribed in elementary school. Analysis of programs, educational works and institutional discourse]. *Colloque Didapro-Didastic 8*. https://hal.univ-lille.fr/hal-02462385/document

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. https://doi.org/10.1007/978-94-011-5470-3_15

Vergnaud, G. (2009). The theory of conceptual fields. *Human Development, 52*(2), 83–94. https://doi.org/10.1159/000202727

Vermersch, P. (2006). Les fonctions des questions [The functions of questions]. *Expliciter, 65*, 1–6. https://expliciter.fr/IMG/pdf/Les_fonctions_des_questions.pdf

Webb, M., Davis, N., Bell, T. Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K–12 school curricula of the 21st century: Why, what and when? *Education and Information Technologies, 22*, 445–468. https://doi.org/10.1007/s10639-016-9493-x

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. https://doi.org/10.1007/s10956-015-9581-5

Wilensky, U. (1995). Paradox, programming and learning probability. *Journal of Mathematical Behavior, 14*(2), 253–280. https://doi.org/10.1016/0732-3123(95)90010-1

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Affiliations

Chantal Buteau,
Brock University, Canada
Department of Mathematics and Statistics
e-mail: cbuteau@brocku.ca

Laura Broley,
Brock University, Canada
Department of Mathematics and Statistics
e-mail: lbroley@brocku.ca

Kirstin Dreise,
Brock University, Canada
Faculty of Education
e-mail: lbroley@brocku.ca

Eric Muller,
Brock University, Canada
Department of Mathematics and Statistics
e-mail: emuller@brocku.ca