Buteau, Muller, & Ralph (2015). Integration of Programming in the Undergraduate Mathematics Program at Brock University. In Online Proceedings of the *Math + Coding Symposium,* London (Canada), June 2015.

Integration of Programming in the Undergraduate Mathematics Program at Brock University

Chantal Buteau, Eric Muller, & Bill Ralph

In this paper we briefly describe the implementation of a sequence of three undergraduate programming project-based mathematics courses at Brock University (Canada) since their inception in 2001: classroom (lecture and laboratory sessions); curriculum, including a detailed example of a programming-based mathematical task; selection of programming languages; assessment; and a brief account of the course evolution until today. We end by addressing the question: how do we know that "it" works?

KEYWORDS: programming; undergraduate mathematics education; third pillar of scientific inquiry of complex systems; classroom implementation; programming-based mathematical task; assessment; course evolution.

1. Introduction

Since 2001, the Department of Mathematics and Statistics at Brock University has integrated programming into its core undergraduate mathematics program (Ben-El-Mechaiekh, Buteau, & Ralph 2007; Ralph 2001) known as MICA, an acronym for *Mathematics Integrated with Computers and Applications*. Through a sequence of three innovative project-based MICA courses, mathematics majors and future mathematics teachers learn to design and program interactive computer environments, which we have called *exploratory objects* (EOs), to investigate mathematical concepts, conjectures, theorems, or real-world situations (Muller, Buteau, Ralph, & Mgombelo 2009). These courses were designed around two of the program's principle objectives: i) to encourage mathematical creativity, and ii) to develop mathematics concepts hand-in-hand with computers (Ben-El-Mechaiekh et al. 2007). These MICA courses are partly a response to society's need for mathematicians who are proficient in using technology to understand complex systems which the European Mathematical Society (2011) has identified as the third pillar of scientific inquiry: "*[t]ogether 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).

In this paper we briefly describe the implementation of MICA I-II-III courses[1]. Section 2 describes the classroom implementation, i.e., lecture and laboratory sessions, as well as the overall curriculum. In Section 3, we elaborate on the implemented programming-based mathematical tasks, and include a detailed example. We report on the selected programming languages in Section 4, and in Section 5, we briefly describe the assessment used in the MICA courses. Section 6 reports on the evolution of the MICA courses since 2001. We end by addressing the question: *How do we know that "it" works?*

[1] The MICA I-II-III courses that we refer to in this paper are the courses as described in the official 2015-16 Brock calendar (http://www.brocku.ca/webcal/2015/undergrad/math.html), namely MATH 1P40, 2P40, 3P40. See Section 6 for a brief account of the evolution of theses courses since 2001.

2. The MICA classroom: lectures and laboratory sessions, and curriculum

The challenge of learning both mathematics and programming and then putting the two together in a useful way could easily be completely overwhelming to students. In order to ameliorate this problem, the pedagogy of the MICA courses was purposely designed to build students' confidence and fluency by having them create progressively more challenging and interesting mathematics EOs (Buteau & Muller 2014). In first-year, MICA I students start to engage in the "third pillar" by learning computer programming in an accessible and engaging mathematics context (Buteau & Muller 2014). The focus in second-year MICA II and third-year MICA III courses is to code and use simulation to explore more advanced mathematics and investigate more complex systems. Table 1 illustrates the mathematics curriculum covered in the sequence of the three project-based MICA courses by listing the assignments that are central to these courses[2]. Topics in MICA I have remained more or less unchanged since 2001, whereas topics in MICA II-III courses have evolved both over the years and according to the instructor's mathematics interests.

Year	**Project**	**Topic**	
1	EO 1	Conjecture about primes or hailstone sequence	
	EO 2	RSA encryption method	
	EO 3	Discrete dynamical system (cubic with two parameters)	
	EO 4	**Original, end-of-term project**	
Cohorts		***(Ralph's 2011-12 cohort)***	***(Fuks' 2014-15 cohort)***
2	EO 5	Buffon needle problem & Monte Carlo integration	Discrete equations: Model of water pollution in a system of two connected lakes connected by a stream
	EO 6	Stats application to stock market	Systems of discrete equations: models of the spread of infectious diseases
	EO 7	Synchronization of traffic lights	Dynamical system of the logistic function & bifurcation diagram
	EO 8	Markov chains applied to income demographics and chronic illness	Stochastic models of bacterial growth
	EO 9	**Original, end-of-term project**	
3	EO 10	Dynamical system of the logistic function & bifurcation diagram	Cellular automata models of road traffic flow
	EO 11	Simulation of battles (Lanchester equations)	Empirical modes and curve fitting: investigations of the Zipf's law.
	EO 12	Prey-predator biological model (Lotka-Volterra)	ODE models in pharmacokinetics, Michaelis-Menten equation
	EO 13	Cellular automata, simulation of epidemics & costs	Pursuit problems in 2D, Hathaway's circular pursuit
	EO 14	**Original, end-of-term project**	

TABLE 1. Examples of MICA mathematics curricula that have been evolving both in time and according to the instructor's mathematics interests.

[2] For a detailed description, together with screenshots, of all fourteen EO projects of a student who enrolled in MICA I in 2011, see Buteau, Muller, Marshall, Sacristán, & Mgombelo (submitted).

The format of each MICA course is two weekly hours of lecture and two weekly hours of computer laboratory session and course sections are usually capped at 35 students. The lectures mostly provide the mathematical content as background and motivation for the programming-based mathematical tasks done or prepared during the lab sessions. In particular, the MICA I lectures include classroom activities that encourage students to make their own mathematical conjectures which we see as an initial step in the third pillar. Since the MICA I course has a large programming component, the laboratory sessions have been carefully designed to help students create their first mathematics EOs (Buteau & Muller 2014). Up until now, most students have enrolled in MICA I with no prior programing background. Table 2 provides a summary of the lab activities in MICA I which highlights the progression of programming concepts needed and learned throughout the first 10 weeks of MICA I; the last 2 weeks are devoted to the end-of-term EO project.

Week	**Programming concepts**	**Main Lab Activity**	**Assignment submission**
1	Designing a graphical interface	Writing (i.e., copy line by line) a first program	
2	Programming variables	"Hello World" & creating a basic calculator,	
3	Conditional structure, loops	Check primality of an integer	
4	Function and Sub-procedures	Create table of n, n^2, & n^3	EO 1
5	Arrays	Powers in Z_n	
6		gcd, Euler's function, inverse in Z_n	
7	Graphics *(& systemic exploration of a concept)*	Draw different shapes & exploration of Euler's theorem	EO 2
8	Graph of a function	Graphing a parabola, points, and coordinate system	
9		EO for the exploration of the dynamical system of the logistic function (numerical table and cobweb)	
10	*(systemic exploration of a dynamical system)*	Exploration, guided by instructor, of the dynamical system based on the logistic function	EO 3
11		Individual work on final project	
12		Individual work on final project	EO 4 *(in following week)*

TABLE 2. Programming concepts learned through activities during the two-hour weekly MICA I laboratory sessions.

In the second-and third year MICA II-III courses, lectures continue to provide the mathematical content for students to create and use mathematics EOs on a broad-range of topics. They usually each contain five EO projects (programmed in vb.net and at times possibly in Maple). The lab sessions are also used to introduce ancillary material such as reading data from files or the use of multi-dimensional arrays.

3. The main programming-based mathematical tasks – The EO projects

The creation and use of the EO assignments is done in part during laboratory sessions but mostly outside classroom time. Each EO project contains not only the interactive environment, but usually also a written (i.e., a type of scientific lab) report. Among the fourteen individual EO projects that students construct over their three MICA courses, eleven of these are assigned

through guidelines provided to students, and the remaining three, at the end of each course, are on a topic selected by the students themselves either individually or in groups of two or three.

The mathematical work in each assigned EO involves two facets: firstly, students create programs based on mathematics from the lectures, and secondly, they use their EO program to conduct experiments to explore mathematics that is unknown to them (Buteau et al., submitted). In some cases, students are also required to develop the model under investigation, and will need to learn and program mathematics not covered in lectures (Buteau et al., submitted). Diagram 1, summarizes a task analysis for EOs and shows the decision process that follows once students select a topic. It provides some insights into the students' activities as they create and use their EO (Buteau & Muller, 2010; Marshall & Buteau, 2014). We suggest that it provides a possible model of a student engaging in the third pillar of scientific inquiry.

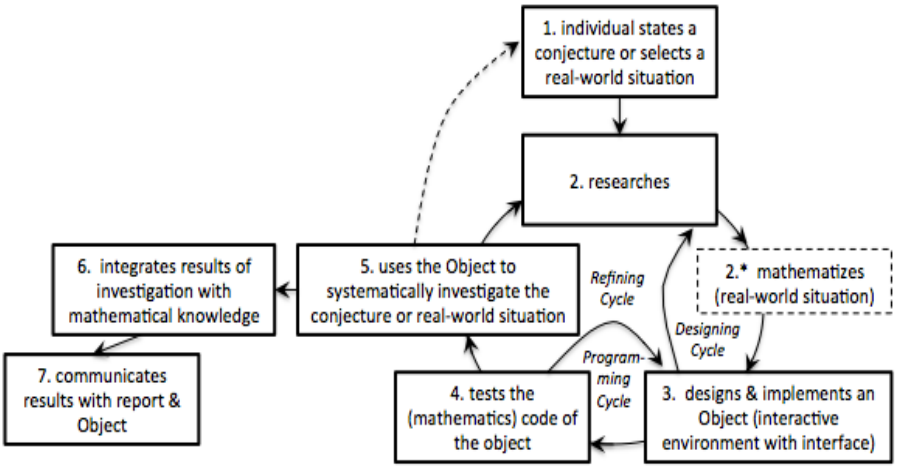


DIAGRAM 1. Development Process Model of a student creating a computer environment for a mathematical investigation or application (modeling or simulation) (Buteau & Muller 2010; Marshall & Buteau 2014).

As an example we consider the first-year EO 3 assignment for which the assignment guidelines given to the students can be found in Appendix 1. During lectures, students use the logistic function to learn about dynamical systems and cobweb diagrams. In the laboratory sessions, students progressively learn about graphical representations of dynamical systems in vb.net over three lab sessions: at first, they explore graphics features in vb.net (this corresponds to week 7 in Table 2), and in the following lab session (week 8 in Table 2), they engage in a guided task about graphing a parabola and a coordinate system – see Appendix 2 for the lab guidelines. In this exercise, students find that they need to known the mathematics behind coordinate changes which motivates the teaching of this material at this point. Situations like this one in which the mathematics is taught after the student encounters a problem or issue often arise in these MICA courses. In the third lab session, students code the graph of the logistic function which requires a parameter, generate the orbits and create the cobweb diagram (this corresponds to week 9 in Table 2). The following lab session involves an interactive lecture in which students use their individual EOs created in week 9 to conduct an instructor guided exploration of what happens to the logistic dynamical system as the parameter is changed. Finally students modify, extend, and use their code to explore the system based on a cubic function involving two parameters and submit their EO and their write up as their EO 3 assignment. A screenshot of one student's approach to this third assignment is shown in Figure 1.

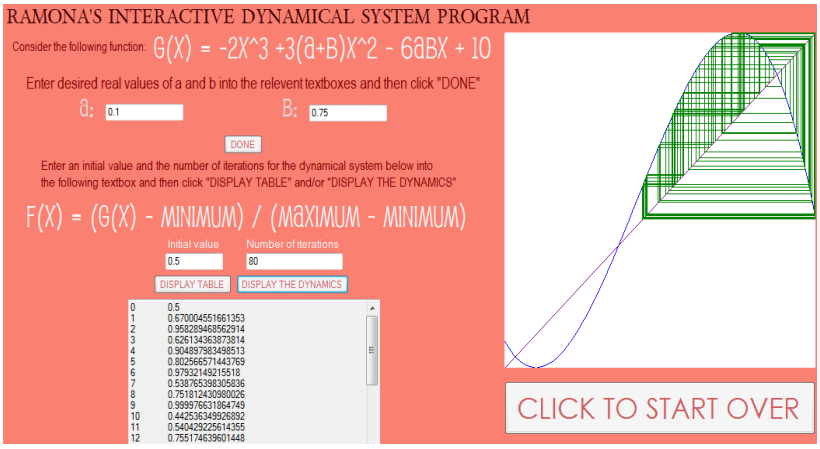


Figure 1. Screenshot of a student's EO 3 for the exploration of the dynamical system based on a cubic (Buteau et al., submitted).

Each MICA course culminates with an original end-of-term EO project (EOs 4,9,14 in Table 1) for which students are encouraged to select a topic of interest to them. Future teachers may decide to create an EO for the step-wise guided learning of a school mathematics concept, which we have called a *learning object* (Muller et al. 2009). Examples of original EO and learning object projects can be found on MICA URL (n.d.); for example, MICA students Matthew and Kylie wondered if it is better to walk or run in the rain (Figure 2, left), while Adam investigated the bounded area, as the exponent increases, of the iterative complex function defining the Mandelbrot set (Figure 2, right). These original projects "*can be regarded as milestones where students are able to demonstrate their ability to engage in the third pillar on a topic of their choice*" (Buteau et al. submitted, [p.15]).

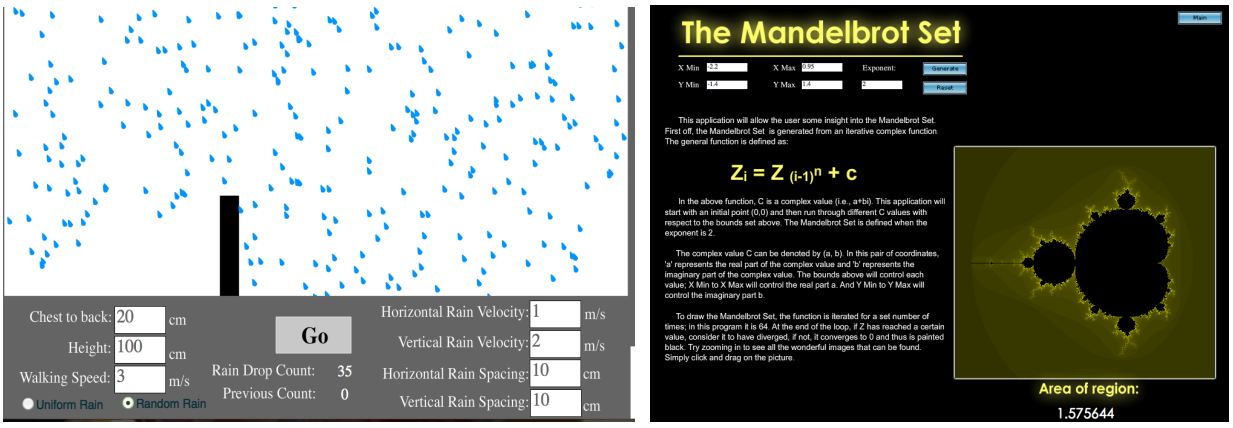


FIGURE 2. To the left, Matthew and Kylie's real-world situation original EO project: "*Is it better to walk or run in the rain?*"; to the right, Adam's pure mathematics EO project about the bounded area of the iterative complex function defining the Mandelbrot set as the exponent increases. See MICA URL (n.d.) to run these EOs or for other examples of students' original EO projects.

4. Selected programming language(s) in MICA courses

Since 2002, Visual Basic.NET has been used in the MICA I course, and continued in MICA II. Sometimes the instructor of MICA III changes to Maple rather than continuing to use vb.net. There are two main reasons for using vb.net, namely: i) vb.net is being widely used in industry for rapid prototyping; and ii) for its user-friendliness, in particular for the design and use of graphical interfaces.

Our students have the option of taking two additional elective programming-based courses focussed on partial differential equations (MATH 3P51 *Applied Mathematics with Maple* and MATH 3P52 *Partial Differential Equations in C++*). The former usually involves programming Maplets in Maple, whereas the latter involves C++ language.

Furthermore, many of our students voluntarily decide in their third or fourth year to enrol in a formal programming course (e.g., JAVA) offered by the Computer Science department. The rationale they most often mention is that they: i) seek greater fluency with programming for their mathematical activities; and ii) want enrich their portfolio for future employability purposes.

5. Assessment in MICA courses

The activity of creating and using mathematical EOs is central to the MICA courses and its importance is reflected in each of the MICA course's evaluation scheme where about 75% of the total grade is assigned to these projects. The remaining 25% is allocated to written tests that focus on the mathematical content of the course. In the MICA I course we've also added, since 2012, short programming quizzes to motivate the students and to try to ensure that they have learned enough of the programming basics for them to succeed in the subsequent MICA courses.

In regard to the evaluation of the projects, each one is graded according to specific criteria that assess the EO and written report that focuses on mathematics and considers communication, effectiveness, interface design, programming sophistication and creativity. See Appendix 1 for an example of the grading scheme outline for EO 3 assignment.

6. The continuing evolution of MICA courses

Originally, the MICA courses were introduced as core courses, spread across 5 terms, for all mathematics majors and future mathematics teachers: 1 term in year 1, a full second-year course, and a full third-year course (Buteau, Muller, & Marshall forthcoming). These courses were in fact introduced in the context of the new core undergraduate mathematics program, also called MICA, launched in 2001 by our department (Ralph 2001). The MICA courses described in the departmental adoption document in 2000 have however evolved over time (Buteau et al. forthcoming).

In the first implementation of MICA I course in 2001, the Department required its first year students to complete a JAVA course prior to MICA I. It seems that these mathematics students found this course challenging because it did not include any programming relevant to mathematics (Muller et al. 2009). In 2002 the Department removed the JAVA course requirement and integrated VB.net programming within the MICA I course. This has made a big improvement in student attitudes and this change has remained until today.

In 2009, the third full-year MICA core course lost its MICA designation and was split into two elective term courses focused on PDEs, namely, MATH 3P51 and MATH 3P52 already mentioned in Section 4 (Buteau et al. forthcoming). The MICA courses now occupied three terms, one in first year, and a full-year (two terms) in second year.

In 2015 our department decided to rename its core undergraduate program as *B.Sc. Mathematics and Statistics* and made modifications to the core curriculum. The second full-year MICA course was split into two one-term courses to be offered in year 2 and year 3 (i.e., MICA II and MICA III in this paper[3]) of the renamed B.Sc. program. Furthermore, MICA I-II courses have remained in this new program a core requirement for all majors except those opting for the pure mathematics concentration, whereas MICA III remains a core requirement for only one concentration, namely the *mathematics integrated with computers and application* (MICA) concentration, as well as for the B.Sc. Mathematics/Co-op program. Students enrolled in the mathematics education concentration, or in the B.Sc.(math)/B.Ed program, are now required to complete a MICA III* course, titled *Visual and Interactive Mathematics*, for which MICA II course is a prerequisite and which includes a focus on learning objects. The newer version of MICA III and MICA III* courses will be offered for the first time in 2016-17.

7. Conclusions - How do we know that "it" works?

This important question was raised during the discussion following the Buteau presentation at the Symposium and having taken some time to reflect, we have decided to respond in more detail in the Proceedings. Our reflections are based on our experience of 15 years of implementation and what we have learned from our research. In the following, we answer the question, *how do we know that "it" works?*, by considering "it" from three different perspectives of MICA courses: i) the curriculum; ii) the pedagogy used; and iii) the students' learning experiences.

i) An important addition to a traditional mathematics curriculum

If 'it' is the MICA curriculum, then we need to be aware that course curricula, in many university mathematics departments, including the one at Brock, are approved by a majority of the full time faculty. Traditionally faculty spend most of the time focused on the mathematical content including pre-requisites of a new or revised curriculum and they spend little or no time considering the delivery of that curriculum. In the case of the MICA courses three aspects raised the most discussion: the first arose from the non-traditional nature of the mathematics content (for example simulation and modelling) which was to be covered in the first three years of the undergraduate degree; the second was the designation of the courses as core, that to be required of all majors, and; the third focused on the need for students to program or code. After much study the Department approved the MICA courses and these have been offered continuously since 2001. So the curriculum of the MICA courses received from the department's faculty as much, if not more, scrutiny as other core courses in analysis, algebra, etc. The revisions to the MICA courses, in 2015, were also scrutinized in the same manner, and the fact that the Department decided to continue them in a revised format indicates that the courses have worked for the past 14 years.

[3] The MICA courses in this revised B.Sc. mathematics program will be taught starting Fall 2015. The description of the MICA courses in this paper represents their implementation since 2002.

Our confidence that the MICA curriculum worked was reinforced by a much larger body of mathematicians when in 2011 the European Mathematical Society identified a third pillar of scientific inquiry of complex systems that highlighted "*a combination of modeling, simulation, optimization and visualization*" (p.2), areas of mathematics integrated 10 years before in the MICA courses. Another key part of the MICA curriculum, programming, was also scrutinized by the department's faculty. Indeed, our confidence that programming in the MICA curriculum works was also reinforced by a much larger body, namely the Society for Industrial and Applied Mathematics (2012) which indicated that "*programming and computer skills are the most important technical skills that new [mathematician] hires take to their jobs*" (p.25). The integration of programming into our mathematics curriculum aligns with recent changes in national school curricula, e.g. England, France, and Finland (Misfeldt & Ejsing-Duun 2015; SITRA 2014). In summary, we argue that the MICA course curriculum works as it embodies approaches of experimental mathematics, inquiry-based learning, learning by using/modifying mathematics simulation, and learning mathematics by programming (Marshall & Buteau 2014).

ii) A different pedagogy that effectively integrates technology into the teaching of mathematics

If 'it' is the pedagogy used in the MICA courses, we need to be aware of the difference between the pedagogy in many traditional university mathematics courses and the pedagogy in the MICA courses. Although different pedagogies have been developed in specific mathematics courses, the most common method of university mathematics teaching is one where the instructor provides and explains the concepts and techniques through lectures and the student is expected to work through these together with additional instructor prepared problems. Normally settings with assistance are provided such as, tutorials, laboratories, help sessions, etc. What is atypical in the MICA courses? First the pedagogy aims to empower students to raise their own mathematical conjectures or raise a question about a real-world situation which they feel can be analysed quantitatively. Second the pedagogy aims to empower students to design, program, and use interactive computer environments, to explore their own mathematical conjecture or real world problem. In our research (e.g., Buteau et al. forthcoming) we have identified a close parallel between the pedagogy used in the MICA courses with the constructionism paradigm (Papert 1991). Therefore not only do we rely on the instructors and TA's views and impressions that, the MICA pedagogy works but our confidence is also bolstered by some of the research results in the constructionism and microworld literature (e.g., Papert 1980, 2000; Hoyles & Noss 1992).

iii) MICA course learning experiences favourably reviewed by students

If 'it' is the students' learning experiences in MICA courses, we turn to students' work, their engagement in the classroom, and their views of how their MICA experience has worked for them. As Instructors, we have not only observed during lab sessions how students have been engaging in the programming-based mathematical activities (EOs), but have also evaluated their mathematics content tests as well as their EOs which overall provide us with some confidence of students' learning in MICA courses. As researchers, we thoroughly examined the complete mathematical work of a student over the 16 months of her MICA I-II-III courses (Buteau et al. submitted). Her experience suggests a greater retention and deeper knowledge of the covered mathematical content in MICA courses, as well as proficiency in the third pillar due to the continuum of programming found throughout the sequence of the three-term MICA courses. The student herself reflects (Muller, Buteau & Sacristán 2015, p.217):

> MICA is the reason why I've succeeded during my coop work terms. It has provided me with both the mathematical background and programming knowledge required to exceed management expectations.

Another student seems to point not only to a sense of proficiency in engaging, with programming, in the third pillar, but also to a sense of epistemological empowerment (Buteau et al. submitted, [pp.14-15]):

> 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.

Overall, students seem to also view developing, as they progress through their MICA courses, 15 key competencies similar to those observed from research (e.g. Wilensky 1995) on programming-based, constructionist approaches to mathematics learning; for example i) to engage in the process of mathematics research; and ii) to closely reflect on problems (Buteau, Muller, & Marshall 2014). However, we are aware that some students find it more difficult to engage in learning in the MICA courses. For example, a student indicates: "Sometimes I was not even able to understand these mathematical models because I was too focused in getting the code right. It was a very stressful course" (Buteau et al. forthcoming, [p.10]).

In a recent paper (Muller et al. 2015) discussing the MICA implementation in *Mathematics Today*, the professional journal for the UK *Institute for Mathematics and Applications*, we have included solicited reflections by students who had completed the MICA courses. We end with some of these reflections: Laura, a student now completing her masters in mathematics, indicates:

> [l]earning how to program and then use that skill to 'do math' in this new sense gave me a feeling of empowerment as a young mathematician that I had never felt previously.

Jessica, a high school mathematics teacher and programme leader for her school board explains:

> The MICA program truly deepened my understanding of mathematics in the current world. This understanding has been beneficial to me in my work with secondary students as I can explain how mathematical modeling is applicable to the world around them and even help them with the basics of modeling real-world phenomena using computer-based applications.

Colin, a post-doctoral fellow in a school of pharmacy concludes our examples of students' reflections:

> During these [MICA] courses, the wide breadth of computational tools that were used to solve mathematical problems granted me with an increased capability to tackle problems from many different research areas. The exposure to computer programming has also proven invaluable to my development as a scientist.

References

Ben-El-Mechaiekh, H., Buteau, C. and Ralph, W. (2007) MICA: A Novel Direction in Undergraduate Mathematics Teaching. *Canadian Mathematics Society Notes*, 39 (6), 9-11.

Buteau, C., & Muller, E. (2010). Student Development Process of Designing and Implementing Exploratory and

Learning Objects. *Proceedings of the sixth conference of European Research in Mathematics Education,* Lyon, France, 2009, 1111-1120.

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 Netherlands.

Buteau, C., Muller, E., & Marshall, N. (2014). Competencies Developed by University Students in Microworld-type Core Mathematics Courses. Proceedings of the *Joint Meeting International Group Psychology Mathematics Education (PME 38)*, 209-18.

Buteau, C., Muller, E & Marshall, N. (forthcoming): When a university mathematics department adopted core mathematics courses of unintentionally constructionist nature – really? Accepted for publication in *Digital Experience in Mathematics Education*.

Buteau, C., Muller, E., & Marshall, N., Sacristán, A.I., & Mgombelo, J. (forthcoming). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Submitted manuscript.*

European Mathematical Society (2011). *Position Paper of the European Mathematical Society on the European Commission's Contributions to European Research* [online]. http://ec.europa.eu/research/horizon2020/pdf/contributions/post/european_organisations/european_mathematical_society.pdf. Accessed 14 September 2015.

Hoyles, C., & Noss, R. (1992). A pedagogy for mathematical microworlds. *Educational studies in Mathematics*, *23*(1), 31-57.

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.

Marshall, N., Buteau, C., & Muller, E. (2014). Exploratory Objects and Microworlds in university mathematics. *Teaching Mathematics and its Applications, 33*, 27-38.

MICA URL (n.d.). *Exploratory and Learning Objects created by Brock students in mathematics courses* [Online]. Available: www.brocku.ca/mathematics/studentprojects

Misfeldt, M., & Ejsing-Duun, S. (2015). Learning mathematics through programming: an instrumental approach to potentials and pittfalls. In *Proceedings of the 9th Congress of European Research on Mathematics Education.*

Muller, E., Buteau, C., Ralph, W., & Mgombelo, J. (2009). Learning mathematics through the design and implementation of Exploratory and Learning Objects. *International Journal for Technology in Mathematics Education*, 16(2), 63-74.

Muller, E., Buteau, C., & Sacristán, A.I. (2015). Through the Looking-Glass: Programming Interactive Environments for Advanced Mathematics. *Mathematics Today* (Dec. 2015), 212-217.

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

Papert, S. (1991). Situating Constructionism. In I. Harel, & S. Papert (Eds.), *Constructionism* (1–11). Norwood, NJ: Ablex Publishing Corporation.

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. IBM Systems Journal, 39.

Ralph, W. (2001). Mathematics takes an exciting new direction with MICA program. *Brock Teaching,* 1(1), 1.

SITRA – the Finnish Inovation Fund (2014). Future will be built by those who know how to code. Retrieved from http://www.sitra.fi/en/artikkelit/well-being/future-will-be-built-those-who-know-how-code

Society for Industrial and Applied Mathematics (2012). *Mathematics in Industry Report*. Retrieved from http://www.siam.org/reports/mii/2012/report.pdf

Wilensky, U. (1995). Paradox, programming and learning probability. *Journal of Mathematical Behavior*, 14(2), 231–280.

Buteau, Muller, & Ralph (2015). Integration of Programming in the Undergraduate Mathematics Program at Brock University. In Online Proceedings of the *Math + Coding Symposium,* London (Canada), June 2015.

Appendix 1 (EO 3 Assignment Guidelines — Winter 2014)

Your goal is to write a program that allows a user to first input a particular cubic equation and then explore its dynamics on the interval [0,1].

Participation in the lecture [about the interactive exploration of the dynamical system of the logistic function] with a *working* program**: (5 marks)**

Part I: Your interactive dynamical system exploration program on a CD (or USB key)

1) The user should be able to enter the parameters a and b for the function $g(x)=-2x^3 + 3(a+b)x^2 - 6abx + 8$

2) Your program should (internally) find the exact maximum M and minimum m of the function g(x) on the interval [0,1]. *(Hint: Use the closed interval method)*

3) Set f(x) = (g(x)-m)/(M-m). Note that the range of f(x) is exactly [0,1]. At the click of a button, we see the graphs of y=f(x) and y=x appear. (These graphs should just touch the bottom and top of your picture box.) **(20 marks)**

4) The user should be able to enter an initial value for the dynamical system determined by f and at the click of another button see a table of values appear and the dynamics (cobweb) drawn in the picture box as in parts 6) to 11) of lab#9. **(25 marks)**

Your program should have an attractive user-friendly interface and good programming style: it should use comments, functions and sub procedures, and should be efficient. **(10 marks)**

Part II: The exploration and hand-written (or typed) report. Your hard-copy report will consist of four parts under the following headings:

1. INTRODUCTION. Write a short paragraph introducing your project. If you use resources (internet, book, article, etc.), give the reference(s) — up to 8 lines **(2 marks)**
2. MATHEMATICS BACKGROUND USING AN EXAMPLE — up to 2 pages **(8 marks)**
    a. Use the two last digits, d_1 and d_2, of your student number and set the values $a=d_1/10$ and $b=d_2/10$; this defines a specific function g. Use it in the following.
    b. Using any technology (e.g. Maple), draw the graph of g with domain [0,1].
    c. Find the maximum and minimum of g, and define f as in step 3 (Part I).
    d. Using any technology (e.g. Maple), draw the graph of f, and write a sentence or two to explain its relation to the graph of g.
    e. Select an initial values x0, use your program to compute the first 10 terms of the sequence of the iterative function system based on f, and use the data to explain how the sequence is built. Identify the convergence or divergence of the sequence.
    f. Draw manually the corresponding cob-web (in the graph of f), and describe how the convergence or divergence of the sequence is visualized.
    g. Show how to find (algebraically) the fixed points of f (you may use Maple for computations), and plot them in the graph of f. Using your program, classify them (attracting, repelling or neither) and describe in your own words what each classification means.
3. DATA COLLECTION ABOUT INTERESTING CASES— up to half a page
    a. Use your program to find values of a and b so that f(x) has three fixed points. Use any method (including Maple) to prove that they are fixed points. Classify each point as attracting or repelling or neither and give written evidence for your claims. **(20 marks)**
    b. Find 3 different pairs of values of a and b and a starting value so that subsequent values oscillate closer and closer to a finite number (between 3 and 100) of values. Describe what happens. **(7 marks)**
4. DISCUSSION/CONCLUSION. Write a short paragraph concluding your exploratory work (e.g., discuss further about 3a) or 3b); about dynamical systems; about the use of cobwebs, etc). If you use resources (internet, book, article, etc.), give the reference(s) — up to half a page **(3 marks)**

Appendix 2 (Excerpts of Lab 7, 8, &9 Guidelines — Winter 2014)

In the following, we provide excerpts of lab guidelines used in MICA I course in Winter 2014 for preparing students to their third assignment about the exploration of the dynamical system based on a cubic – with two parameters (EO 3 in Table 1).

1. Excerpt of Lab #7 Exploring Euler's theorem & Introduction to Graphics Guidelines

Exercise 1: Introduction to graphics

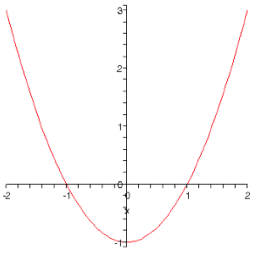
Create a new vb.net project. Put a picturebox and a button on a form and put this code under the button to see some of the graphics commands at work. ***Note that the coordinate system on the picture box puts (0,0) at the upper left hand corner***.

```
Dim g As Graphics
Dim sdBrush As SolidBrush = New SolidBrush(Color.Red)
Dim PBlue As New Pen(Color.Blue)
Dim PRed As New Pen(Color.Red)
Dim i, h, w, a, b As Integer
g = PictureBox1.CreateGraphics
g.DrawLine(PBlue, 23, 56, 200, 300)
g.DrawRectangle(PRed, 23, 56, 20, 30)
g.DrawEllipse(PRed, 150, 150, 250, 50)
g.FillRectangle(sdBrush, 40, 60, 20, 30)
h = PictureBox1.Height   'what is the following code doing exactly?
w = PictureBox1.Width
For i = 0 To 100
        a = Int(w * Rnd())
        b = Int(h * Rnd())
        g.DrawRectangle(PRed, a, b, 1, 1)
Next i
```

2. Lab #8 Function Graphs Guidelines

A) Graph of a Parabola

Create a new project with two buttons (Draw the parabola! & End) and one picture box. When clicking on the 'Draw the parabola' button, the graph of the function $y=x^2-1$ will appear in the picture box. At the end, your graph should look like the following but without the numbers or tick marks on the axes:



a) Create a picture box (600 x 600). Draw the 2 axes in black.
b) Draw the origin as a yellow point
c) Draw 5 blue points on the graph of $y= x^2-1$ for x=-2, -1,0,1, and 2.
d) Write a function

```
Function xc(ByVal x As Double) As Integer
```

that changes the x-coordinates for graphing purposes.
e) Write a similar function called yc for the y-coordinates.

f) Draw the graph of the function in red. ***Hint:*** *If (x1,y1) and (x2,y2) are two consecutive points to be graphed, then graph the line joining these two points to get a smoother graph. Or draw many many points for which the x-values are next to each other.*

B) More Graphs

Create a new project called MyChaos. This project will be completed next week (at that point, you'll understand why it is called that way!). Your interface should contain two buttons ('Compute!' and 'End') and four text boxes.

1) Insert a text box and name it txtk. The value stored in this textbox will be the (real) factor to the quadratic function f(x)=kx(1-x). Declare k as a global variable.

2) Define a function

f(ByVal x As Double) As Double

that returns the value f(x)=kx(1-x).

3) Write code to graph the parabola f(x)=kx(1-x) when the button is clicked, where k is a real number and dom f = [0,1]. The parabola will always start at the lower left of the picture box and end at the lower right of the picture box.

- When k=2 the parabola's maximum should be at the middle of the picture box. When k=4 the parabola's maximum should just touch the top of the picture box.
- Use coordinate change functions for changing the mathematics coordinates to vb.net graphing coordinates (see part A)
- Use a Sub procedure Sub DrawFunctionGraph() to draw the function graph

4) Write a Sub procedure to add the graph of the diagonal y=x

3. Lab #9 Chaos Experiment Plate-Form Lab Guidelines

In this lab we create a program that will allow us to systematically explore next week the dynamical system based on the logistic function. The following explains all the steps. It builds on Part B of lab 8:

[**Note:** Your 'Chaos Experiment Plate-Form' **must** be completed (5% of your Assignment 3 mark) *before* the interactive lecture next week on March 20 during your scheduled lab.]

1) Create a new project, called MyChaos, with an interface that contains a button, a text box, and a picture box. The dimensions of the picture box should be exactly 400 by 400.

2) Add a text box called txtInitial, and label it. The value stored in this textbox will be the initial value used in the dynamical system.

3) Add a text box called txtBound, and label it. The value stored in this textbox will be the number of iterations of the dynamical system.

4) Add a text box called txtk. The value stored in this textbox will be the (real) factor k of the logistic function f(x)=kx(1-x). Define k globally, and define a vb.net function

Function f(ByVal x As Double) As Double

which returns the value kx(1-x).

5) Add a text box called txtOutput. The values that are output into this textbox will contain the entire sequence of values generated by the dynamical system. Set the Multiline property to true and then resize the box to make it as large as will still fit on your form. Set the ScrollBars property to Vertical.

6) Define a sequence of numbers by the formula $x_{n+1}=f(x_n)$ for which f(x)=kx(1-x). Let M be the value the user entered in txtBound. Write code that stores the values $x_0, x_1,..., x_M$ in an array (define the array as global, and have the values computed and saved in the array in a Sub procedure:

Sub ComputeSequence(ByVal m As Integer)

where m is the number of terms computed in the sequence, i.e., x0, x1, ..., xm. Also the following M+1 lines of text appear in txtOutput for which the x_i's are the actual numerical values:

0	x_0
1	x_1
...	...
M	x_M

8) Write code to graph the parabola f(x)=kx(1-x) when the button is clicked, where k is a real number between 0 and 4 that the user has entered into the textbox. The parabola will always start at the lower left of the picture box and end at the lower right of the picture box.
- When k=2 the parabola's maximum should be at the middle of the picture box. When k=4 the parabola's maximum should just touch the top of the picture box.
- Use coordinate change functions for changing the mathematics coordinates to vb.net graphing coordinates (see lab 8)
- Use a Sub procedure Sub DrawFunctionGraph() to draw the function graph

9) Write a Sub procedure to add the graph of the diagonal y=x

10) Let x_0 be the value the user entered in txtInitial. Write code so that a small red filled circle appears at the point (x_0, x_0) on the screen, when the button is clicked.

11) Add more code to complete the cobweb: when the button is clicked, the following pair of lines is drawn on the screen for every i from i=0 to i=M-1 the line from (x_i, x_i) to (x_i, x_{i+1}) **AND** the line from (x_i, x_{i+1}) to (x_{i+1}, x_{i+1}). This should all be done through a Sub procedure:

Sub Cobweb(ByVal M As Integer)

And now experiment with different values of k and different values of x_0 . Can you get a general sense of where this dynamical system is well behaved and where it is chaotic? We will explore the amazing behaviour of this system in detail next week.