Enhancing Practice and Achievement in Introductory Programming with a Robot Olympics

Michael James Scott, Steve Counsell, Stanislao Lauria, Stephen Swift, Allan Tucker, Martin Shepperd and Gheorghita Ghinea.

***Abstract*—Computer programming is notoriously difficult to learn. To this end, regular practice in the form of application and reflection is an important enabler of student learning. However, educators often find that first-year B.Sc. students do not readily engage in such activities. Providing each student with a programmable robot, however, could be used to facilitate application and reflection since, potentially, robots facilitate engaging learning experiences whilst providing immediate and intuitive feedback. This paper explores whether an introductory course centred upon programming personal robots in preparation for an end-of-course event day—a Robot Olympics—can help students to firstly, engage in programming practice more frequently and secondly, improve the quality of their code. A survey was conducted to examine the students' programming practice behaviour and students' final coursework submissions were also reviewed for aspects of program quality. The findings from this cohort were compared to a reference-group from a previous cohort that shared similar learning objectives and entry requirements, yet had focused on web programming as opposed to using robots. The results reveal statistically significant increases in programming practice compared to the reference-group. Furthermore, being enrolled on the course culminating in the Robot Olympics was a significant predictor of two aspects of program quality: functional coherence and sophistication. This suggests that robot-centred courses can promote engagement with, and enhance some aspects of, programming practice.**

***Index Terms*—Personal Robots, CS1, Introductory Programming, Achievement, Practice, Motivation, Quality.**

I. Introduction

REGULARLY writing and evaluating code plays an important role in learning computer programming. This is because ongoing self-regulated practice plays a key role in the transition from novice to expert [1]. It has been reported that ten years of such practice is needed to become a highly regarded software engineer [1], [2]. Consequently, introductory programming courses aim to encourage students to practice frequently and to actively reflect on how they can improve the quality of their code.

Programming, however, has a history of poor outcomes at the introductory level [3] and educators often encounter students whom do not regularly engage with programming tasks [4], [5]. This can occur because programming presents seemingly insurmountable challenges to beginners [6], often leaving them to feel frustrated and confused [7]–[9]. As a result, educators use the laboratory setting to provide feedback and to support students in an attempt to motivate them *as* they encounter challenges [4], [5]. However, such efforts are limited and may not encourage practice beyond the laboratory environment.

It is, therefore, important to examine how effective forms of practice can be promoted beyond the laboratory setting. An approach that shows promise is the use of personal robots [10], whereby each student is provided with their own programmable robot. In theory, they make learning activities more engaging, motivating students to spend more time experimenting with their programs [11]. They also provide a more intuitive source of feedback as they reinforce mental models in a visual way [12], helping students to fix problems and overcome frustrations in a relatively short time.

However, while the potential impact of robots is promising, it is important to situate robots in an appropriate context in order to maximise that impact. That is, robots are tools that support, complement and enhance learning environments. As an example, robots frequently grab attention [11], [13], [14]. Thus, robot can draw upon students' curiosity to engage them with an assignment. However, there is little evidence to suggest that the mere presence of a robot makes an assignment more relevant, better able to inspire confidence, or more satisfying [15]. Therefore, such constructs may need to be addressed through other aspects of the learning environment.

This article evaluates one such environment: a course centred around a Robot Olympics. In this introductory programming course, students learn to program their own personal robots during worksheet-based laboratory sessions. To engage students in practice that leads to improvements in the quality of their code, their experimentation with their personal robot is supported through regular code reviews. This prepares students for an end-of-course event—the Robot Olympics itself—where they program their own robots to complete a specific task in a dedicated space as an assessed demonstration.

II. Related Work

The use of robots in educational contexts has grown in popularity since the early Millennium [16] and robot-centred courses are often positively received by students [11]. Furthermore, a systematic review has shown that robots, in general, can be effective when teaching computer programming [17]. However, there are questions about the effectiveness of robot-centred courses, highlighting a need to explore achievement data [18].

Manuscript received June 30, 2014 and revised December 1, 2014.

The authors are based in the Department of Computer Science, Brunel University London, UK, e-mail: michael.scott@brunel.ac.uk.

Digital Object Identifier 10.1109/TE.2014.2267094

Where such evidence is available, robot-centred programming courses have not consistently demonstrated success. A study conducted at the US Air Force Academy found that scores in a robotics section of a programming course were lower than in a non-robotics section [16]. Limited access to the robots has been suggested as the potential reason for this result, as students received insufficient time to reflect and engage in further experimentation [16]. Consequently, the availability of robots could moderate the effectiveness of a robot-centred course.

Now that low-cost robots such as the Finch [19] are available, students can learn using their own personal robots. In order to motivate students to engage in programming practice with their robots, various strategies can be used [20]. A popular choice is hosting a Robot Olympics [21]–[25]. These 'games' often take the form of events where students demonstrate solutions to a set of well-defined tasks. For example, students in the Trinity College Fire-Fighting Home Robot Contest explore programming through the development of a robot which can navigate a mock-up home to extinguish a candle [24]. This approach would seem to represent a novel and engaging learning environment. However, the method has not been formally evaluated as a model for an introductory programming course, so the potential impact is unclear.

It is important to clarify this impact to help those deciding to introduce robots into a course. For example, Kumar questions "is it worth using robots for traditional projects in [an] AI course?" and concludes "no, if we consider the time and effort that robot projects demand" [26]. Such demands range from storage, locating spare robots when students forget them, diagnosing abnormal robot behaviours, repairing mechanical failures, and the increased duration of assessments; which, may not be justified if any positive impacts are marginal.

III. Intended Outcomes

This article evaluates the impact of a robot-centred programming course, in terms of levels of practice and achievement, through comparison with a reference-group. The reference group was drawn from a previous year on the same programme within the authors' department. Both groups had the same entry requirements and followed a similar structure. As typical for a UK institution, each involved a year-long course structured across two terms of twelve weeks. However, the earlier cohort focused on web programming as the robots had not been introduced at that point into the department. As such, the following research questions are posed in order to determine differences between the two cohorts:

- RQ1. Will the students enrolled on the course leading to the Robot Olympics report spending more hours per week practising their programming skills compared to those enrolled in the web programming course?
- RQ2. Will the students enrolled on the course leading to the Robot Olympics produce code of an overall higher quality compared to those enrolled in the web programming course?
- RQ3. Will greater levels of practice and practice within the context of preparing for a Robot Olympics predict aspects of code quality differently?

As it is possible that robots may not be equally effective for all students, gender differences are also explored. The first research question therefore examines two hypotheses: male and female students enrolled on the robot-centred course will report more hours of practice per week ($H_{1\text{-}2}$). The second research question examines three hypotheses: practice will have a direct effect on overall code quality (H_3), enrolment on the robot-centred course will have a direct effect on overall code quality (H_4) and gender will have a direct effect on overall code quality (H_5). The third research question addresses three core hypotheses: that greater practice, gender, and course enrolment will have different impacts on three different aspects of code quality: functional coherence; readability; and sophistication. As this represents three variables of interest and three aspects of code quality, this results in an additional nine hypotheses ($H_{6\text{-}14}$).

IV. Course Design

Both courses were practical introductions to computer programming which expected students to:

- LO1. Demonstrate an understanding of the basic concepts of programming
- LO2. Analyse a problem and produce a computer program as a solution to that problem
- LO3. Use a simple development environment to produce viable program code

As both cohorts were supervised by a similar team of core teaching staff and the robot-centred course built upon existing teaching practice within the department, both courses followed similar basic structures. This consisted of a series of lectures introducing concepts to students and laboratory sessions which then reinforce those concepts through practical programming tasks (similar to [27]). Laboratory sessions were organised weekly to ensure regular practice. To encourage attendance, the web programming cohort had mid-term examinations while those preparing for the Robot Olympics had their code formally reviewed by teaching assistants. This meant that all students received ongoing soft scaffolding [28] and feedback at regular intervals [29] as this strategy is believed to be more effective for encouraging practice [4], [30].

Both cohorts completed their respective programming tasks as part of a group project. Thus, students were divided into groups of five or six. Meetings with group tutors facilitated individual support and helped to prompt students to reflect on their progress through small group learning activities [31]. This strategy also helps students form learning communities, encouraging mutual support during challenging tasks and transforming experimentation with robots into social learning opportunities [32], [33]. It also reflects and presents (in a small way) the composition and communication issues found in the IT industry [34].

While there are many similarities between the two courses, there is also a number of key differences. Table I presents these differences, showing that the courses are comparable but also highlighting several confounds. The earlier cohort focused exclusively on web programming and the later cohort on the use of the robots. As the learning objectives were

TABLE I
KEY DIFFERENCES BETWEEN THE WEB PROGRAMMING COURSE AND THE ROBOT-CENTRED COURSE

Course Element	Web Programming	Robot Olympics
Personal Robot	✗	✓
JavaServer Page Project	✓	✗
Robot Olympics Project	✗	✓
Assessed Worksheets	✗	✓
Exams	✓	✗
Oral Viva	✗	✓
Python Classes	✓	✗
Java Classes	✓	✓

the same, there were many similarities between the tasks each group member had to undertake: firstly, both had to be written in Java; secondly, both had to demonstrate the same range of programming constructs; thirdly, both examined how students separated the user interface (i.e., presentation layer) from the key functionality (i.e., domain logic layer); finally, both validate user input and both demonstrate file processing. The assessed problems were also designed to be of similar size and complexity. However, the robot coding problems were different in nature as they were designed to capture students' interest and make use of the robots' capabilities[1].

The web form processing tasks, such as adding content to a file and displaying it on a web page, were replaced with 'events' within the robot Olympics. Examples include: Morse code communication, where the robot would use the light on its beak to communicate Morse code translations; obstacle course, where the robot had to navigate across an arena; robot controller, where the movement of the robot is controlled directly through a user interface and tunnel navigation, where the robot measured the lengths of tunnels with its light sensor.

Another key difference is that the web programming cohort followed a combined Python and Java curriculum which focused on JavaServer Pages (JSP) (see [35] for details). As such, new learning content replaced some of the previous material (i.e., python and JSP classes) in order to help students with the new mode of assessment (i.e., using the robots).

V. METHOD

A. Data Collection

For administrative reasons, the sampling frame for each cohort were the students who had completed at least one code review by the end of the final term. A random sampling procedure was used to select participants. Data was collected in three rounds: a paper-based questionnaire was distributed to all students in the laboratory during their final laboratory session; a digital version was then advertised on the virtual learning environment and email alerts were distributed to those whom had not responded to the paper-version; after ten days, an additional series of follow-up emails were distributed to the non-respondents. All participants were offered an explicit opt-out for further communication at each stage.

[1]Further details on the Robot Olympics and its assessment method are available at: http://dx.doi.org/10.13140/2.1.3680.9281

Target sample sizes were calculated using Cochran's formula [36] and adjusted for anticipated non-response. From 126 invitations for the web programming cohort and 115 invitations for the robot-centred cohort, 91 and 84 responded. Thus, response rates were 72% and 73%, respectively, noting that 34 and 30 cases were classified as late because significant follow-up was required to elicit their response.

Data was screened for non-response bias. Firstly, demographic variables were compared to known population characteristics including age, gender and previous experience. There were no significant differences. Secondly, timely respondents were compared to late respondents. Although late respondents reported lower self-efficacy for programming tasks (p = .008), higher programming anxiety (p = .007) and lower interest in programming (p = .000), there were no significant differences in practice or achievement.

B. Participant Characteristics

Participants were first-year undergraduate students following "Computer Science" or "Business Computing" at the authors' institution[2]. In the web programming cohort, the average age was 19.5 and 26.5% of the respondents were female. Approximately 46.8% reported no prior programming experience, with 26.6% having a vocational or high school qualification in computing. In the robot-centred cohort, the average age was 19.6 and 16.8% of the respondents were female. Similarly, 45.5% reported no prior programming experience, with 31.2% having a vocational or high school qualification in computing. There were no statistically significant differences in terms of age or gender. Additionally, there was no statistically significant difference in terms of prior experience.

C. Research Instruments

1) Self-Reported Weekly Programming Practice: Hours of programming activity per week was assessed using a self-report measure. The item "in a typical week during term-time, how frequently did you write code and/or work on programming related activities?" was presented as a 7-point Guttman-style item. Each response option was labelled "at least {} hours" increasing in multiples of five.

2) Code Quality: Quality of coursework submissions was scored according to a marking scheme by a single rater. Although tasks were different, both shared a common set of learning objectives and level of sophistication. Three aspects of code quality were assessed: *functional coherence*, which measured whether solutions successfully implemented the set requirements; *readability*, which measured whether the solution was commented and structured appropriately for future maintenance; and *sophistication*, which measured whether an appropriate range of programming constructs had been used. Each aspect was scored according to five descriptors, with the lowest indicating inadequate quality. Moderation of 12 truncated submissions showed that self-consistency ($\alpha = .91$) and faculty agreement with marks ($\alpha = .72$) were adequate [37].

[2]http://www.brunel.ac.uk/cedps/computer-science/undergraduate-studies

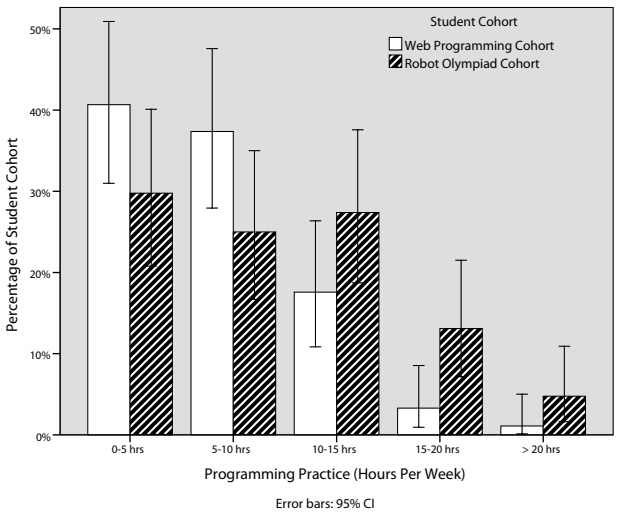

Fig. 1. A clustered bar chart comparing self-reported hours of programming practice between students enrolled in the Web Programming and Robot-Centred courses.

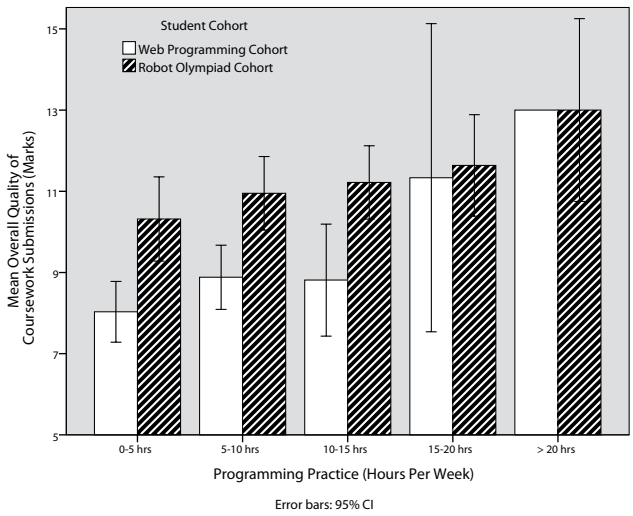

Fig. 2. A clustered bar chart comparing the overall quality of final coursework submissions between students enrolled in the Web Programming and Robot-Centred courses at each level of self-reported practice.

VI. Data Analysis

The data was analysed using PASW 18.0.3 for Windows. Cases with missing data were excluded list-wise. All *p*-values from null-hypothesis significance tests are two-tailed with statistical significance being determined at the conventional level (α = .05). Table III on the following page provides a summary of adjustments for multiple hypotheses testing using the Benjamini-Hotchberge Procedure [38].

A. Greater Practice with the Robot Olympics (RQ1)

As self-reported programming practice did not follow a normal distribution, a Mann-Whitney U Test was conducted to examine the difference between the two cohorts. This indicated that programming practice was greater in the robot-centred course compared to the web programming course for both male students ($U = 1404, p = .016, r = 0.21$) and female students ($U = 73, p = .024, r = 0.40$). This is shown in Figure 1, where the proportion of students studying for less than 10 hours per week decreases and those studying for more than 10 hours per week increases. This represents an overall increase of 37.4% based on the mean difference. However, it should be noted that the median did not change and 54.8% did not fulfil the expectation of 10-15 hours per week.

B. Higher Overall Quality with Practice and the Robot Olympics (RQ2)

A factorial (2 x 2 x 5) between-subjects MANOVA evaluated the impact of the robot-centred course at each level of practice that students reported that they engaged in on the quality of students' code submissions. Assumptions of normality were supported. However, Box's and Brown-Forsyth tests only supported equality of variance and equality of the covariance matrices once readability was collapsed into four categories (rather than five). As the cell sizes were not equal, Pillai's Trace was used to assess significance. Significant multivariate effects for practice ($Trace = .190, F = 2.202, p = .011, \eta_p^2 = .063$) and for course ($Trace = .134, F = 6.603, p < .001, \eta_p^2 = .134$) were found. However, there were no significant effects for gender ($Trace = .019, F = 0.829, p = .480, \eta_p^2 = .019$).

TABLE II
ANOVA Results For Each Aspect of Code Quality

Predictors	F	p	η_p^2	d
Functional Coherence				
Level of Practice	5.321	.001	.141	
Course	19.268	.000	.129	1.15
Gender	—	—	.003	0.27
Readability				
Level of Practice	0.919	.455	.028	
Course	1.504	.222	.011	0.36
Gender	—	—	.016	0.39
Sophistication				
Level of Practice	0.798	.529	.024	
Course	5.426	.021	.040	0.74
Gender	—	—	.011	0.37

Note: The estimate of Cohen's d was calculated using the estimated marginal means from the MANOVA and the pooled standard deviation of each variable.

C. Varying Effects of Practice and the Robot Olympics on Aspects of Code Quality (RQ3)

To examine the effects on each aspect of code quality, univariate ANOVAs were conducted. These are shown in Table II above. As no gender differences were found in the multivariate test, no hypotheses tests were conducted and only the observed differences are shown. The results suggests that preparation for the Robot Olympics helped students produce higher quality code in terms of functional coherence and sophistication. Furthermore, programming practice predicted functional coherence. Interestingly, however, programming practice itself did not predict higher code readability or sophistication.

VII. Discussion

The findings reinforce the notion that robot-centred learning environments can encourage programming practice. The frequency of students reporting at least 10 hours of practice per week increased from 22% to 45%. Nevertheless, engagement remains an issue with more than 50% of students not pursuing levels of practice set out as an expectation. Further investigation into student practice could result in improvements to the Robot Olympics by, for example, considering other potential influences (see [4]).

The results reveal some evidence which suggests that practice within the context of the robot-centred course can be more effective than alternatives. This is an important to consider, because *how* students practice is just as, if not more, important than how long they practice for [1]. Examining overall quality ratings across different levels of practice revealed that those involved in the Robot Olympics consistently outperformed those on the web programming course at lower levels of practice. However, further qualitative enquiry is needed to explain how. This is because the available data cannot isolate the contribution of any individual change to the course, such as the use of personal robots.

Enrolment on the robot-centred course predicted functional coherence, even when accounting for practice as a covariate. This suggests that students fulfilled the requirements more successfully. Hence, introducing personal robots within an appropriate context can lead to improvements in student outcomes in a way that just additional time-on-task does not. An insight is that the physical feedback can be easily understood by students, whereas a small difference in a web page may not be so noticeable. Thus, the way the nature of the coding interface and its feedback supports students' development of mental models warrants investigation.

It was anticipated that the code reviews would help students improve the readability of their code. However, enrolment on the robot-centred course did not predict readability. There are several explanations for this, with one such hypothesis being that students prioritised the functionality of the code as watching robots complete tasks is more compelling. However, the format of the reviews could also be a factor.

Enrolment on the robot-centred course predicted sophistication where practice did not. As such, the challenges presented during code reviews and the robots themselves could have pushed students to improve. It is interesting to note that the increase was supported by a higher proportion of those engaged in low levels of practice receiving higher scores. Perhaps differences in style of cognition, creative thinking, and reflection promoted this increase. Further work is required to isolate and explore these hypotheses.

There were no statistically significant differences in terms of gender. However, there were some potentially meaningful differences in terms of effect size. Most notably, female students showed greater changes in level of practice based on course enrolment than male students. Additionally, there could be small but meaningful differences in terms of achievement, but this cannot be verified due to the low statistical power associated with *post-hoc* analyses.

TABLE III
Summary of Findings and Adjusted P-Values

RQ	H_n	Hypothesis	$\tilde{p}$	Conclusion
1	H_1	$\Delta PRACT \neq 0$ $(GEN = male)$	.035	*Supported*
	H_2	$\Delta PRACT \neq 0$ $(GEN = female)$	.037	*Supported*
2	H_3	$PRACT \rightarrow QLTY$	.030	*Supported*
	H_4	$BOT \rightarrow QLTY$	.000	*Supported*
	H_5	$GEN \rightarrow QLTY$	.528	—
3	H_6	$PRACT \rightarrow FC$	.003	*Supported*
	H_7	$BOT \rightarrow FC$	.000	*Supported*
	H_8	$GEN \rightarrow FC$	—	—
	H_9	$PRACT \rightarrow RD$	.528	—
	H_{10}	$BOT \rightarrow RD$	.305	—
	H_{11}	$GEN \rightarrow RD$	—	—
	H_{12}	$PRACT \rightarrow SO$	.529	—
	H_{13}	$BOT \rightarrow SO$	.037	*Supported*
	H_{14}	$GEN \rightarrow SO$	—	—

Note: Univariate tests for gender have been excluded as no multivariate significance was found in H5; $\tilde{p}$: Benjimini-Hochberg adjusted p-value; PRACT: Self-Reported Hours of Programming Practice Per Week; QLTY: Overall Code Quality; BOT: Enrolment in Robot-Centred Course; GEN: Gender; FC: Functional Coherence of Code; RD: Readability of Code; SO: Sophistication of Code.

VIII. Limitations

The study was observational in nature, so confounds may exist. For example, the two cohorts could have differed at baseline on unobserved variables, there may have been differences in teaching quality, etc. Furthermore, as there were several changes, only the entire course is assessed rather than individual differences. The analysis itself focuses on quantitative data, excluding potentially useful qualitative data on the benefits and challenges associated with the robots. As such, it is unclear whether their characteristics encouraged different approaches to learning and writing code. Caution should be taken when interpreting self-report questionnaire data [39]. Additionally, only those whom submitted code for review were included, perhaps excluding those that failed to engage with the robots, and the *post-hoc* hypotheses associated with gender had a high probability of type-II error.

IX. Conclusions

Practice and reflection both play important roles in the development of programming expertise. As such, it is important to design courses that encourage these activities. As summarised in Table III, this article explores a robot-centred approach designed to promote these activities which was trialled in an actual course used at the authors' institution. This reveals some evidence that the use of personal robots in an appropriate context can inspire students to engage in frequent practice. Enrolment on the new course also predicted two aspects of code quality: functional coherence and sophistication. This demonstrates that the robot-centred course improved student outcomes in a way that just additional time on task does not, suggesting a qualitatively different learning experience. For example, did the physical feedback from the personal robot aid in the construction of mental models? As such, further work is needed to evaluate how students' reflective activities, meta-cognition, creativity, attitudes, motivation, and approach to learning changed as a result of participation in the new robot-centred course.

REFERENCES

[1] K. A. Ericsson, R. Krampe, and C. Tesch-Römer, "The role of deliberate practice in the acquisition of expert performance," *Psychological Review*, vol. 100, no. 3, pp. 363–394, July 1993.

[2] L. E. Winslow, "Programming pedagogy–a psychological overview," *SIGCSE Bulletin*, vol. 28, no. 3, pp. 17–22, Sep. 1996.

[3] M. Guzdial, "From science to engineering," *Commun. ACM*, vol. 54, no. 2, pp. 37–39, Feb. 2011.

[4] M. J. Scott and G. Ghinea, "Educating programmers: A reflection on barriers to deliberate practice," in *Proceedings of the 2nd HEA Conference on Learning and Teaching in STEM Disciplines*, Birmingham, UK, April 2013, pp. 85–90.

[5] T. Jenkins, "Teaching programming: A journey from teacher to motivator," in *Proceedings of the 2nd HEA Conference for the ICS Learning and Teaching Support Network*. London, UK: AIED, July 2001, pp. 53–58.

[6] ——, "On the difficulty of learning to program," in *Proceedings of the 3rd HEA Conference for the ICS Learning and Teaching Support Network*. Loughborough, UK: HEA, July 2002, pp. 1–8.

[7] P. Kinnunen and B. Simon, "Experiencing programming assignments in cs1: the emotional toll," in *Proceedings of the Sixth international workshop on Computing education research*, ser. ICER '10. New York, NY, USA: ACM, 2010, pp. 77–86.

[8] C. Rogerson and E. Scott, "The fear factor: How it affects students learning to program in a tertiary environment," *Journal of Information Technology Education*, vol. 9, no. 1, pp. 147–171, 2010.

[9] N. Bosch, S. K. D'Mello, and C. Mills, "What emotions do novices experience during their first computer programming learning session?" in *Proceedings of the 16th International Conference on Artificial Intelligence in Education*. Memphis, TN: AIED, July 2013, pp. 11–20.

[10] J. Kay, "Robots in the classroom...and the dorm room," *Journal of Computing Sciences in Colleges*, vol. 25, no. 3, pp. 128–133, 2010.

[11] S. A. Markham and K. N. King, "Using personal robots in cs1: Experiences, outcomes, and attitudinal influences," in *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2010, pp. 204–208.

[12] D. B. Adams, R. Louis, B. Morin, J. Cerrato, J. Keidel, J. Vincent, J. Merrill, D. Rampelli, K. Gieskes, S. Fellows *et al.*, "Explore-create-present: A project series for cs," in *Proceedings of the ASEE North Central Sectional Conference (ASEE10)*, 2010.

[13] T. Lauwers, I. Nourbakhsh, and E. Hamner, "Csbots: design and deployment of a robot designed for the cs1 classroom," in *ACM SIGCSE Bulletin*, vol. 41, no. 1. ACM, 2009, pp. 428–432.

[14] M. Apiola, M. Lattu, and T. A. Pasanen, "Creativity and intrinsic motivation in computer science education: experimenting with robots," in *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. ACM, 2010, pp. 199–203.

[15] M. M. McGill, "Learning to program with personal robots: Influences on student motivation," *ACM Trans. Comput. Educ.*, vol. 12, no. 1, pp. 4:1–4:32, Mar. 2012.

[16] B. S. Fagin and L. Merkle, "Quantitative analysis of the effects of robots on introductory computer science education," *ACM Journal of Educational Resources in Computing*, vol. 2, no. 4, pp. 1–18, December 2002.

[17] L. Major, T. Kyriacou, and O. Brereton, "Systematic literature review: teaching novices programming using robots," *IET software*, vol. 6, no. 6, pp. 502–513, 2012.

[18] D. Alimisis, "Educational robotics: Open questions and new challenges," *Themes in Science & Technology Education*, vol. 6, no. 1, pp. 67–71, 2013.

[19] T. Lauwers and I. Nourbakhsh, "Designing the finch: Creating a robot aligned to computer science concepts," in *AAAI Symposium on Educational Advances in Artificial Intelligence*, 2010.

[20] E. Roberts, "Strategies for encouraging individual achievement in introductory computer science courses," in *ACM SIGCSE Bulletin*, vol. 32, no. 1. ACM, 2000, pp. 295–299.

[21] R. R. Murphy, "Using robot competitions to promote intellectual development," *AI magazine*, vol. 21, no. 1, p. 77, 2000.

[22] D. Cappelleri and N. Vitoroulis, "The robotic decathlon: Project-based learning labs and curriculum design for an introductory robotics course," *Education, IEEE Transactions on*, vol. 56, no. 1, pp. 73–81, Feb 2013.

[23] B. Ladd and E. Harcourt, "Student competitions and bots in an introductory programming course," *Journal of Computing Sciences in Colleges*, vol. 20, no. 5, pp. 274–284, 2005.

[24] I. M. Verner and D. J. Ahlgren, "Robot contest as a laboratory for experiential engineering education," *ACM Journal on Educational Resources in Computing*, vol. 4, no. 2, p. 2, 2004.

[25] D. Cappelleri and N. Vitoroulis, "The robotic decathlon: Project-based learning labs and curriculum design for an introductory robotics course," *Education, IEEE Transactions on*, vol. 56, no. 1, pp. 73–81, Feb 2013.

[26] A. N. Kumar, "Three years of using robots in an artificial intelligence course: Lessons learned," *J. Educ. Resour. Comput.*, vol. 4, no. 3, Sep. 2004. [Online]. Available: http://doi.acm.org/10.1145/1083310.1083311

[27] R. Geitz, "Concepts in the classroom, programming in the lab," vol. 26, no. 1, pp. 164–168, 1994.

[28] K. D. Simons and J. D. Klein, "The impact of scaffolding and student achievement levels in a problem-based learning environment," *Instructional Science*, vol. 35, no. 1, pp. 41–72, 2007.

[29] G. Gibbs and C. Simpson, "Conditions under which assessment supports students learning," *Learning and teaching in higher education*, vol. 1, no. 1, pp. 3–31, 2004.

[30] J. P. Barros, L. Estevens, R. Dias, R. Pais, and E. Soeiro, "Using lab exams to ensure programming practice in an introductory programming course," vol. 35, no. 3, pp. 16–20, 2003.

[31] L. Springer, M. E. Stanne, and S. S. Donovan, "Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis," *Review of educational research*, vol. 69, no. 1, pp. 21–51, 1999.

[32] I. Verner, "Characteristics of student engagement in robotics," in *Intelligent Robotics Systems: Inspiring the NEXT*, K. Omar, M. Nordin, P. Vadakkepat, A. Prabuwono, S. Abdullah, J. Baltes, S. Amin, W. Hassan, and M. Nasrudin, Eds. Springer Berlin Heidelberg, 2013.

[33] J. E. Froyd and M. W. Ohland, "Integrated engineering curricula," *Journal of Engineering Education*, vol. 94, no. 1, pp. 147–164, 2005.

[34] M. Shepperd, "Group project work from the outset: An in-depth teaching experience report," in *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*. IEEE, 2011, pp. 361–370.

[35] A. Jayal, S. Lauria, A. Tucker, and S. Swift, "Python for teaching introductory programming: A quantitative evaluation," *ITALICS*, vol. 10, no. 1, pp. 86–90, 2011.

[36] J. E. Barlett, J. W. Kotrlik, and C. C. Higgins, "Organizational research: Determining appropriate sample size in survey research," *Information Technology, Learning & Performance*, vol. 19, no. 1, pp. 43–50, 2001.

[37] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[38] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 289–300, 1995.

[39] S. I. Donaldson and E. J. Grant-Vallone, "Understanding self-report bias in organizational behavior research," *Journal of Business and Psychology*, vol. 17, no. 2, pp. 245–260, 2002.

Michael James Scott is pursuing a Ph.D. within the Department of Computer Science at Brunel University London, where he received a B.Sc. in computer science in 2010 and an M.A. in digital games theory and design in 2011. His research explores computer science education with a focus on the impact of games and immersive media experiences.

Steve Counsell is a Reader in the Department of Computer Science at Brunel University London. He received a Ph.D. in software engineering from Birkbeck College, University of London in 2002.

Stanislao Lauria is a Lecturer in the Department of Computer Science at Brunel University London. He received a Ph.D. in cybernetics from the University of Reading in 2001.

Stephen Swift is a Lecturer in the Department of Computer Science at Brunel University London. He received a Ph.D. in intelligent data analysis from Birkbeck College, University of London in 2002.

Allan Tucker is a Senior Lecturer in the Department of Computer Science at Brunel University London. He received a Ph.D. in artificial intelligence from Birkbeck College, University of London in 2001.

Martin Shepperd is a Professor of Software Technologies and Modelling and is Head of the Department of Computer Science at Brunel University London. He received a Ph.D. in measurement theory from the Open University in 1991.

Gheorghita Ghinea is a Reader in the Department of Computer Science at Brunel University London. He received a Ph.D. in computer science from the University of Reading in 2000.