Visualising the code: are students engaging with programming at Level 1?

Elaine Thomas (Project leader)

Soraya Kouadri Mostéfaoui

Helen Jefferis (Consultant)

Contact point: elaine.thomas@open.ac.uk

Submitted: 19th July 2019

Keywords: programming teaching, visual programming, computer science education, student engagement

Contents

CONTENTS 2

Executive Summary **3**

Introduction **3**

Aims and scope of the project **4**

Activities **4**

Literature survey **4**

The case study: TU100 *My digital life* **4**

Findings **9**

Conclusions and future work **10**

List of deliverables **10**

Figures and tables **11**

References **11**

Statement of ethical review **12**

Executive Summary

This project *Visualising the code: are students engaging with programming at Level 1?* investigated the impact of using a visual programming environment on student engagement with programming.

Programming is a subject that many students find difficult and it may be particularly challenging for distance learning students working largely on their own. Many ideas have been put forward in the literature to explain why students struggle with programming, including: the relative unfamiliarity of computer programming or 'radical novelty' (Dijkstra, 1989), cognitive load (Shaffer, 2004) and that the whole learning environment may be influential (Scott & Ghinea, 2013).

We used as our case-study TU100 *My digital life* which is a Level 1 undergraduate Computer Science module in the Open University. The rationale for this work stems from the need for an introductory undergraduate Computing and IT module that will engage students of widely differing levels of prior experience in terms of programming and of education generally. In TU100, the module team introduced a visual programming environment, based on Scratch (MIT, 2007), called 'Sense' which is used in conjunction with an electronic device, the SenseBoard.

In the first phase of the project we analysed the grades of 6,159 students in the final assessment across six presentations of the module to identify student performance in the programming task, in comparison with their overall performance on the module. The aim was to explore whether there was any difference between student engagement with the programming task in comparison with non-programming tasks. Our results suggest that there is no significant difference in levels of engagement between these tasks, and it appears that success, or otherwise, in one type of task is a good predictor of engagement with the other tasks.

In the second phase of the project we analysed the textual comments made by students in the Student Experience on a Module (SEaM) survey from two recent presentations of TU100, using key words relating to programming. Just under 30% of students who made textual comments gave feedback about Sense or the programming teaching. A total of 22.2% of the students made positive comments about the use of Sense or the programming teaching generally and 7.6% of students' comments were negative. Of the students who made negative comments, a small number had struggled with the programming, while others thought that the teaching was pitched at too low a level. However, the majority of student comments in this area suggest that they had enjoyed the programming elements.

The visual programming language used at Level 1 has been successful in engaging students in the study of programming. This study will provide a firm basis for a similar analysis of student performance on the new Level 1 modules which use a visual programming language in the first module followed by Python in the second one, and how well students cope with Level 2 programming.

Introduction

The *Visualising the code: are students engaging with programming at Level 1?* project investigated the impact of using a visual programming environment on student engagement with programming. There is considerable debate about effective methods of introducing programming to students at all levels of education. Programming is a subject that many students find difficult and it is particularly challenging for distance learning students working largely on their own. In recent years, the teaching of programming skills in schools has been seen as a solution to the 'skills gap' in the Information Technology (IT) sector. Yet, student cohorts in the Open University are diverse and many students have been out of formal education for a long time. Therefore, there is likely to be a strong need for the teaching of introductory programming at Level 1 in the University's Computing and IT degree programme for the immediate future.

In this report we will first present the aims and scope of the project followed by a brief overview of the literature around the teaching and learning of programming to explain some of the difficulties that students experience. Then we set out our research methodology and analysis of the results, followed by our main findings and conclusions from the project.

Aims and scope of the project

The aim of this project was to investigate the impact of using a visual or graphical programming environment on student engagement with programming. Our case study is the Level 1 module TU100 *My digital life* which was presented twice-yearly from October 2011 and was presented for the last time in February 2017.

This project addressed the fundamental question as to whether the visual programming environment actually engages novice programmers in a distance learning context.

Specifically we aimed to discover whether students were engaging more with the visual programming environment in TU100 *My digital life* than with the text-based programming language used in the previous Level 1 module, M150 *Data, computing and information* in the final assessment.

Activities

The project first looked at the of current literature on programming and also other work being carried out in the University (such as eSTEeM projects) and held informal discussions with other eSTEeM project teams about their scholarship work on Level 1 programming. In addition there were discussions with the chair of M150 *Data, computing and information* around student engagement with the programming question in the EMAs for that module.

The major activity of the project was the collection and analysis of data from our case study module, TU100 *My digital life*.

Literature survey

We examined a range of existing work on the teaching of introductory programming to adults. This included a brief literature survey on the use of visual programming languages and other environments to teach mature students to gain overview of field.

There is considerable debate about effective methods of introducing programming to students at higher education level. Mostly, the studies are concerned with the teaching of programming in a traditional university setting. Dropout rates for first year students in computing subjects are high across the HE sector in the UK (Lee, 2017) and many students find learning to program difficult (Jenkins, 2002). There are different interpretations as to the reasons for this, perhaps because different skills and types of learning are involved. Programming demands a high abstraction level and generalised way of thinking. Programming also requires a good level of both knowledge and practical problem-solving techniques; as well as practical and intensive study skills. Dijkstra (1989) contends that learning to program entails ‘radical novelty’ as novices may not have acquired the necessary prior skills on which to build and progress. Jenkins (2002) argues that the reason behind the difficulties in learning to program is the blend of learning types required: surface learning for remembering features such as syntax and order of precedence, and deep learning in the understanding of concepts and development of true competence. Shaffer et al (2004) suggest that utilising cognitive load theory (Sweller, 1988) can aid the teaching of novice programming in introductory Computer Science modules. The whole learning environment may be an influential factor so ‘soft scaffolding’ strategies, detailed feedback on work and motivational practices are most likely to be effective (Scott & Ghinea, 2013).

The case study: TU100 *My digital life*

Our case study, TU100 *My digital life*, is a Level 1 undergraduate module worth 60 credits and is part of a common stage 1 curriculum which is taken by all Computing and IT students. As the OU does not require any previous educational attainment as an entry condition, TU100 assumes only a basic level of computing experience. It covers a broad range of computing and networking topics with an emphasis on ubiquitous computing. Programming activities are woven through the teaching material; however, they are only one aspect of the overall study and assessment.

Previous teaching of programming at level one in a 30 credit module used a text-based programming environment, JavaScript, but half of students avoided answering the question on

programming in the End of Module Assignment (EMA). Feedback from students and tutors suggested that JavaScript was unpopular with students (Richards & Woodthorpe, 2009).

Therefore, when TU100 was developed the module team decided to use a visual programming environment the teaching. While a visual programming environment has limitations in terms of employability, it allows students to learn fundamental skills and concepts and build their confidence to enable them to succeed with other programming languages. Therefore, Sense (Open University, 2011), a version of Scratch, was specially developed for the module. Alongside the Sense programming environment, students used the SenseBoard, a small programmable hardware device that can be connected to the student's computer using a USB connection.

TU100 is designed to be delivered by distance learning and is presented in a combination of printed and online self-study materials including resources such as audio and video recordings. TU100 has two presentations per year, each spread over nine months, with the October presentation followed by a February one. Although numbers of students on the module are large, the student experience is very different from a MOOC (Massive Open Online Course). At the OU students have a named tutor who supports them by offering face-to-face and/or online tutorials using an audio-conferencing system and also moderates online forums. Although distance learning tends to be a rather solitary activity, online asynchronous communication tools, such as forums, emails, blogs and wikis are used for communication between students and their peers, their tutors and the module team.

Engagement with course materials in general and programming in particular presents challenges for distance learning students as they may have little recent experience of formal education and they may be working full or part-time whilst studying, and some may have also caring responsibilities. In addition, there were concerns amongst academic staff that students who struggled with the programming could avoid the programming assessment but still pass the module by simply working harder on the other elements of the module.

Data collection and analysis

Data collection and analysis was carried out in two phases:

Phase 1 Acquisition of End of Module Assignment (EMA) data for TU100 from a number of presentations to compare student performance in the Sense programming question with their performance in the other non-programming questions and their overall performance.

Phase 2 Acquisition of Student Experience on a Module (SEaM) data for two recent presentations of TU100 in order to analyse students' textual comments for responses about the programming teaching and learning.

Phase 1 EMA scores data

We collected copies of the EMAs for six presentations of TU100 (October 2013, February 2014, October 2014, February 2015, October 2015 and February 2016) in order to identify the Sense programming task in each EMA (this varied from presentation to presentation).

Then we obtained the question scores and final grades for the students who submitted their EMA in each of the six presentations giving a total of 6,159 student results. Firstly, we identified student performance on the programming task in each EMA. Then the combined results from the six presentations were analysed using SPSS (v22) in order to compare student performance on the programming task with their performance on the non-programming tasks and on the EMA as a whole. We also looked to see if there was any correlation between the results.

Results of the EMA data analysis

The graph in Figure 1 shows the comparison between the mean scores for the programming, the non-programming elements and the average result overall for each of the six student cohorts. The red line shows the number of students (right hand axis) on each presentation.

Quantitative analysis indicates a strong ($r=-0.543$, $p<0.01$) correlation between the scores that students achieved in the programming and non-programming elements of the final assessment, i.e. students who did well in programming also tended to do well in the non-programming

elements, and vice versa. Therefore, results from the statistical data analysis suggest that there is no significant difference in levels of engagement between these tasks, and it appears that success, or otherwise, in one type of task is a good predictor of engagement with the other task.

	Oct 13	Feb 14	Oct 14	Feb 15	Oct 15	Feb 16
Overall Result Mean	70.66	69.62	73.63	73.19	70.03	70.28
Programming Mean	71.32	69.64	85.00	84.10	75.14	73.26
Non-programming Mean	70.52	69.63	70.79	70.46	68.75	69.54
Number Students	1340	801	1343	767	1234	674

Figure 1 Comparison of mean scores for the programming question and the scores for the other non-programming questions, for the six presentations

Phase 2 SEaM survey data

We obtained the SEaM survey results for students on the two most recent presentations of TU100, October 2015 and February 2016. These presentations were also used in the statistical analysis of EMA scores.

In addition to closed questions, the SEaM survey provides open-ended questions where students may make their own textual responses:

- What aspects of teaching materials, learning activities or assessment did you find particularly helpful to your learning?
- What aspects of teaching materials, learning activities or assessment did you find not particularly helpful to your learning? We would welcome any further suggestions or comments to consider for future editions of the module.
- Do you have any other comments to add about your study experience on this module?

In total there were 440 respondents to the SEaM survey across both presentations. Of these 325 respondents had provided textual responses. A form of content analysis was applied to the textual comments (Mayring, 2000, Krippendorff, 2004) based on whether the respondent had used term such as ‘Sense’, ‘SenseBoard’, programming’ and their variants. Initially, responses were coded using NVivo software according to whether they were related to ‘Sense’ or ‘programming teaching’ and whether the views were positive or negative. Once this step was complete, all the responses in these categories were filtered out for further analysis.

During the next stage, further analysis was conducted on the responses in each category to identify any emergent themes. These categories were also quantified to gain an impression of how widespread these views were amongst survey respondents.

Results

The results of the initial analysis indicated that out of the 325 respondents who made textual comments, there were 96 (29.5%) respondents who made unprompted comments which could be categorised as relating to 'Sense' or to 'programming teaching'. Of the responses about the programming, 72 (22.2% of all responses) were categorised as positive and 24 (7.4% of all responses) were categorised as negative. Figure 2 illustrates the proportion of comments about Sense or programming in relation to those unrelated to the programming teaching.

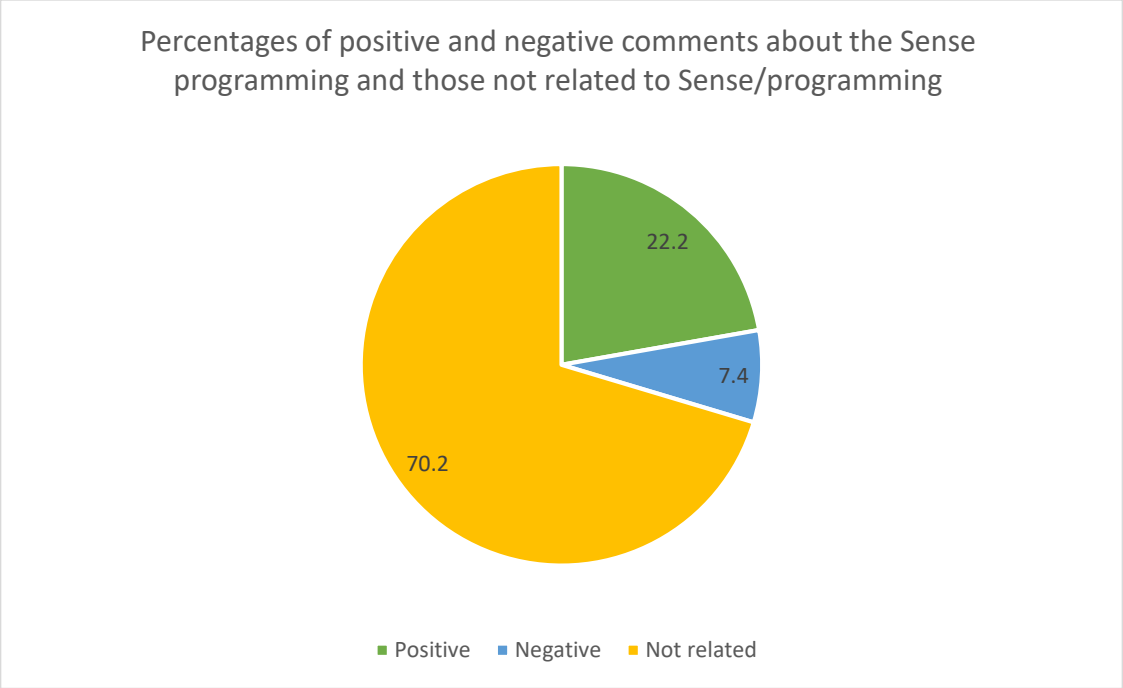

Figure 2 Percentages of responses about programming teaching in relation to all responses

Table 1 shows the distribution of responses on the programming teaching according to the main term used and whether these were categorised as 'positive' or 'negative'. The figures are also expressed as percentages of the overall number of respondents who provided textual comments.

Table 1 Categorisation of comments as positive or negative

Terms	Positive	Percentage of total responses	Negative	Percentage of total responses
Sense	61	18.8%	18	5.5%
Programming teaching	11	3.4%	6	1.9%
Total responses positive/negative	72	22.2%	24	7.4%

The 'positive' category

Further content analysis carried out on the 'positive' category (72 respondents) identified various themes. These are listed below followed by the number of respondents that mentioned them in brackets:

Enjoyment of the Sense/programming work (25)

Helping understanding of programming (11)

Enjoyment and valuing of the practical activities (8)

Good introduction to programming (6)

Helpful Programming guide (6)

The students who made positive comments on the Sense programming teaching also tended to comment positively on the study materials:

Positive experience of the study materials (25)

Online materials (16)

Books (13)

Clarity of materials and instructions (4)

First OU module (4)

Figure 3 shows a word cloud with the most frequently occurring words used in the positive comments.

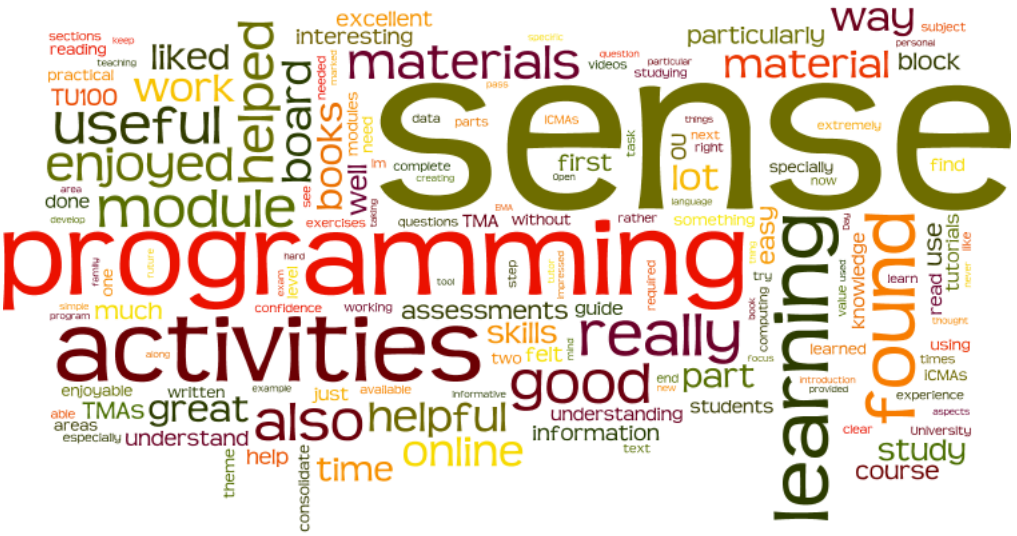

Figure 3 Frequently occurring words in positive comments

The ‘negative’ category

Two sub-categories of negative comments were identified through content analysis of the ‘negative’ comments (24 respondents):

Students who struggled with the programming (4).

Students who thought that the programming teaching was pitched at too low a level (20).

Of the 20 students who thought that the teaching was pitched at too low a level, a range of themes emerged. These are listed below followed by the number of respondents that mentioned them in brackets:

Sense was a good way to start the module as an introduction to programming or a refresher, but the module should have moved on to languages such as Java, JavaScript or Python (6)

Programming teaching is too basic/easy (5)

Recognition that the teaching is aimed at a wide range of students (3)

Sense was enjoyable but not useful for employability (2)

Sense was enjoyable but ‘real’ coding would be better (2)

Sense is not appropriate for an undergraduate module (2)

Some students who made negative comments on the Sense programming teaching also commented on the module as a whole, although opinions varied. For example, one student thought that they had learned a great deal, just not what they had expected, while another student simply disliked the entire module.

Table 2 Examples of 'positive' and 'negative' textual comments from students

Positive comments
I enjoyed the Sense materials, and thought this was a very clever way of introducing what is a very complex subject to beginners. This has sparked an interest for further specialized study in the future.
There were no aspects that were found to be unhelpful. Everything had its function and relevance. I would have enjoyed more Sense aspects, though this is only a personal preference since programming is part of my ultimate goal.
However, my favourite part of the course was programming with Sense. I enjoyed it so much that I would strongly consider taking more advanced programming modules with the OU in the future.
Negative comments
Found the sense very time consuming and needed further in depth studying
I felt that a programming language like python would have been better to use than sense, I do understand that this module was really an introduction to computing and there for sense made sense, but from a personal point of view it did not feel right, due to drag and drop aspects of it, in saying that though I would not dismiss the possibility that I will use it again outside my studies.
Sense. I understand the Open University must be open to people from all backgrounds and I would even be fine starting to code with blocks. But the fact we actually do that for a whole academic year became frustrating beyond words….

Findings

The results of our analysis of the combined results of n=6159 students in the EMA over six presentations of the module suggest that there is a strong correlation between performance in the programming element and the non-programming elements of the Level 1 undergraduate distance learning Computing and IT module.

In addition, the majority of students who mentioned programming in their textual comments in the SEaM survey expressed positive views: they had enjoyed the programming, thought it was a good introduction to programming and helped their understanding of programming. Some of the minority of students who expressed negative views had struggled with the programming on the module while others thought that the programming teaching was pitched at too low a level. In addition, more students have engaged with the programming element than on previous modules as evidenced by the numbers of students who completed the Sense question in the EMA.

Given the diversity of the student cohort on TU100 and, indeed, all Open University modules, the predominance of positive comments in the SEaM survey suggests that the programming teaching is serving a very useful purpose. However, future module design could take account of those students who are ready for more challenging work.

This study has its limitations in that we do not know whether the programming task had any influence on the students who did not submit this final assessment and so failed the module. Also, more detailed work is needed to investigate how the small number of students who did not engage with the programming task but passed TU100 fared in later, more advanced, programming modules. However, the results from this project suggest, that using a visual

programming environment as part of a broader Computing and IT course can be very successful in encouraging students to engage with programming.

Conclusions and future work

In this study, we looked at how Level 1 Open University students performed in their assessment after they had been taught how to program with a visual programming environment (Sense). We also investigated whether students' performance in programming was significantly different from their performance with the non-programming tasks in the final assessment (EMA). A statistical analysis of the students' EMA scores over six different presentations suggests that overall there was not a significant difference in performance between the two elements, neither for the cohort as a whole nor for the individuals, but rather that success or otherwise in one is a good predictor of performance in the other.

We have analysed a large data set of student assessment results and combined our findings with the results of analysis of student textual feedback in two recent SEaM surveys. The quantitative analysis of data from EMAs suggested that concerns that students could avoid the programming but still pass the module were unfounded. A visual programming environment appears to help students to engage with programming being taught by distance learning as shown by the unprompted positive references to Sense and programming in the SEaM textual comments.

We conclude, therefore, that using a visual programming environment has been successful in engaging students in the programming tasks, in comparison with a previous module that used a text-based language.

A similar visual programming environment called OUBuild, also based on Scratch, has been developed at the Open University and is now being used in a new Level 1 module. The previous 60 credit curriculum has been split between two modules designed to be studied in sequence. The first module uses OUBuild to introduce a range of programming concepts, problem-solving and testing and debugging. This module is followed by a module that uses Python, a text-based programming language to extend students' programming skills along with other Computing and IT skills and knowledge. This study will provide a firm foundation and a framework for monitoring students' performance for both the programming and non-programming elements of the new modules.

Impact

The project began its work when the design of the new Level 1 Computing and IT curriculum had already been decided and production was underway. However, the main impact of this project is to establish that the School of Computing and Communications decisions on the design and development of the new Level 1 curriculum were well-founded. The project has confirmed that our impressions of the variation in students' responses to the teaching of programming in TU100 *My digital life* were correct. Therefore, the project will provide a reference point for future studies on the teaching of programming at Level 1 in the School.

In addition to disseminating the work of project within the University, we were able to present our work at an international conference in Zagreb and so promote the School's approach to the teaching of programming to an international audience.

List of deliverables

Thomas, E., Kouadri Mostéfaoui, S. and Jefferis, H. (2017) Investigation of student engagement with programming in TU100: The impact of using a graphical programming environment? *6th eSTEeM Annual Conferenc*e, The Open University, Milton Keynes, 25-26 April 2017. (Presentation)

Thomas, E.; Kouadri Mostéfaoui, S. and Jefferis, H. (2018) Visualising the Code: An investigation of student engagement with programming in TU100. *7th eSTEeM Annual Conference*, The Open University, Milton Keynes. (Presentation)

Thomas, E.; Kouadri Mostéfaoui, S. and Jefferis, H. (2018) Visualising the code: A study of student engagement with programming in a distance learning context. (Presentation)

Thomas, E.; Kouadri Mostéfaoui, S. and Jefferis, H. (2018) Visualising the code: a study of student engagement with programming in a distance learning context. In: *Proceedings of the 11th International Conference on Networked Learning 2018* (Bajić, M.; Dohn, D. N.; de Laat, M.; Jandrić, P. and Ryberg, T. eds.), Springer.

Figures and tables

Figures

Figure 1 Comparison of mean scores for the programming question and the scores for the other non-programming questions, for the six presentations 6

Figure 2 Percentages of responses about programming teaching in relation to all responses..... 7

Figure 3 Frequently occurring words in positive comments 8

Tables

Table 1 Categorisation of comments as positive or negative..... 7

Table 2 Examples of 'positive' and 'negative' textual comments from students..... 9

References

Dijkstra, E.W. (1989) A Debate on Teaching Computer Science: On the Cruelty of Really Teaching Computer Science. Communications of the ACM, 32, 12, 1398-1404. http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html (accessed 19th July 2017).

Jenkins, T. (2002) On the Difficulty of Learning to Program. In Proceedings of the 3rd Annual HEA Conference for the ICS Learning and Teaching Support Network, 1-8. (accessed 19th July 2017).

Krippendorff, K. (2004) *Content analysis: An introduction to its methodology* (2nd ed.). Thousand Oaks, CA: SAGE

Lee, G. (2017) Which universities have the highest first year dropout rates? Channel 4 FactCheck 28th August 2017. https://www.channel4.com/news/factcheck/which-universities-have-the-highest-first-year-dropout-rates (accessed 19th January 2018).

Mayring, Philipp (2000). Qualitative Content Analysis. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research, 1(2), Art. 20, http://nbn- resolving.de/urn:nbn:de:0114-fqs0002204 (accessed 18th July 2019).

Open University (2011) Sense [Computer program]. Available at http://sense.open.ac.uk (accessed 30th September 2017).

Richards, M. & Woodthorpe, J. (2009). Introducing TU100 'My Digital Life': Ubiquitous computing in a distance learning environment. In Ubicomp 2009, 30 Sep - 3 Oct 2009, Orlando, Florida, USA. http://oro.open.ac.uk/26821/2/ubicomp_final.pdf (accessed 19th July 2017).

Scott, M. A. & Ghinea, G (2013) Educating programmers: A reflection on barriers to deliberate practice https://www.heacademy.ac.uk/knowledge-hub/educating-programmers-reflection-barriers-deliberate-practice (accessed 10th July 2017).

Shaffer, D., Doube, W. & Tuovinen, J. (2003) Applying Cognitive Load Theory to Computer Science Education. In M. Petrie & D. Budgen (Eds) Proceedings of the Joint Conference of the European Association of Science Editors (EASE) & the Psychology of Programming Interest Group (PIGG), Keele, UK (accessed 26th September 2017).

Sweller, J. (1988) Cognitive load during problem-solving: Effects on learning. Cognitive Science 12 (2) pp257-285.

Statement of ethical review

Because we were using student data from EMA score analysis and also student comments from SEaM surveys, the project team checked with the University's Human Participation and Materials Ethics Committee (HPMEC) about whether approval is needed for this activity, but we were advised that approval was not needed. All the SEaM data was anonymised by removing students' personal identifiers before the analysis was carried out.