Identifying the challenges of learning programming at undergraduate level

A Threshold Concept approach

Executive Summary

Lucy Yeomans, School of Education, Communication & Society
Steffen Zschaler, Informatics Department
Kelly Coate, King's Learning Institute

Identifying the challenges of learning programming at undergraduate level: A Threshold Concept approach

Background

Programming is an essential skill for any computer science student, indeed many would argue that most STEM (Science, Technology, Engineering and Maths) programmes would expect their students to have some understanding and capability in programming as part of their course (McCracken et al., 2001). Nevertheless, learning to program is generally acknowledged to be very difficult; students are required to have the correct abstract understanding of a concept and be able to implement it in a concrete manner using appropriate strategies (Robins et al., 2003; Lahtinen et al., 2005). This requires a significant amount of hands-on programming experience. However, it has been argued that novice programmers typically have a superficial understanding of programming that is context specific and therefore they struggle with knowledge transfer (Lahtinen et al., 2005). Furthermore, the heterogeneity of student cohorts regarding their experience of programming makes differentiation extremely problematic and this is often attributed as one of the major factors contributing to high drop-out rates on university courses (Jenkins & Davy, 2002; Lahtinen et al., 2005; Pedroni et al., 2009).

Threshold Concepts

The notion of 'Threshold Concepts' has been used in wider education literature to identify concepts that are central to students' mastering of a particular subject area and that, it is suggested, should be a new focus of teaching (Rountree & Rountree, 2009). Threshold Concepts can be defined as core concepts which are particularly difficult to overcome. Meyer and Land (2005) describe them as theoretical summits that, once reached, signify either a leap forward in an individual's understanding, a clarity of a concept's complexity and how it connects to other ideas, and/or a point in which a significant idea becomes embedded within someone's knowledge in such a way that it would be hard to undo. They subsequently simplified this definition into four characteristics: *Troublesome*, *Transformative*, *Integrated* and *Irreversible*.

Since Meyer and Land's initial proposal the uptake of a Threshold Concept approach to teaching and learning in Higher Education has been enthusiastic. Nevertheless, the definition and process of identification of Threshold Concepts remains subjective and contested (Barradell, 2012). Firstly, because there is a lack of consensus as to how many of the four characteristics are required to make a concept a 'threshold', rather than simply a 'core' concept (Barradell, 2012). Furthermore, Davies (2006) concedes that Threshold Concepts may be additionally difficult to identify because within any given discipline they may be 'taken for granted' and as such will not be made explicit. Rountree and Rountree (2009) argue that the grasping of Threshold Concepts will rarely be achieved during one eureka moment, but instead take place over a transitional period of time - from one state of being and knowing to another. According to Threshold Concept theory, such transitional periods, or 'liminal spaces', are where students are most likely to get stuck.

The study

Much like other institutions, the undergraduate students in the Informatics Department at King's College London are extremely diverse in terms of their experience and proficiencies at programming; cohorts have increased significantly year on year while supervision capacity remains constant. In response to the challenges this presents, a study was devised to explore how identifying relevant Threshold Concepts could support more effective pedagogical approaches to programming education for first year undergraduate students.

Methodology

The study used a qualitative interview approach, shaped by a theoretical framework based on Threshold Concept theory. The project team decided initially to focus on the first two characteristics of Threshold Concepts suggested by Meyer and Land (2005) – *troublesome* and *transformative*. Research instruments were designed with those in mind, with questions related to aspects of programming that participants have/had struggled with as well as which had potentially provided them with a paradigm-shift in understanding of the subject matter.

Further investigation of the Threshold Concept literature suggested additional dimensions of the *troublesome* characteristic to be considered in the study: students may use 'mimicry' as a way of grappling with difficult concepts, reproducing what they have been shown without concrete understanding - although this can be viewed as, for some, a useful step towards more complete understanding of a concept (Meyer & Land, 2005; Hughes & Peiris, 2006; Rountree & Rountree, 2009). Eckerdal et al. (2007) identified a framework to be used in identifying when and where a student is in a liminal space while learning to program. The framework was based on different sorts of conceptual understanding: abstract/theoretical understanding of a concept; concrete understanding of the concept evidenced through practical programming; the ability to go from abstract to concrete understanding; understanding why the concept is used and taught and understanding the application of the concept in new situations. The final consideration was the emotional response associated with liminal spaces and programming in particular: an emotional reaction to a concept was regarded by Rountree and Rountree (2009) be an indicator of its potential to be a threshold concept, both frustration and elation could show at which point a student is within liminal space. Additional dimensions of the *transformative* characteristic taken into account included the proposal that once students master a Threshold Concept they can be said to be acquiring a new identity, that of an 'insider' within a discipline rather than a student practising computer science (Eckerdal et al., 2007; Rountree & Rountree, 2009). This notion of 'feeling like a programmer' was also explored in the study.

In an attempt to achieve some consensus regarding the Threshold Concepts for novice student programmers, data was gathered from three sample populations: first year undergraduate students, third year undergraduate students and professional programmers with a range of programming experience. The rationale was to capture a sense of which concepts were identified at the beginning and end of the undergraduate experience, as well as incorporating the perspectives of practitioners far more established in the field. This was a response to the identification of a gap in Threshold Concept literature related to stakeholders outside the immediate learning environment (Rountree & Rountree, 2009; Barradell, 2012).

Methods

Observations and unstructured interviews with first year undergraduate students participating in pair-programming labs took place over a three-month period, where students were asked questions regarding their experiences of programming and which aspects of the course they enjoyed and didn't enjoy. While these data were not extensively used in the analysis, they formed a crucial step in informing the questions asked for the next data collection phase and also, on a practical note, to improve student participation rates when they were invited to take part in focus groups designed to investigate potential Threshold Concepts in a more focussed manner. Again participants were asked particularly about aspects of programming they found challenging, but they were also asked to discuss any concepts that changed their perception of programming itself. The introductory questions were then followed by an activity whereby participants were asked, as a group, to physically organise a list of concepts covered in their curriculum in order of difficulty. The final arrangement was then used as a prompt in further questioning related to potential Threshold Concepts as discussed above.

Focus groups were also held with students in their third year, where they were asked to consider both their experiences in their first year but also their current views, and with professional programmers who came from a range of training backgrounds and amount of experience. The industry professionals were additionally asked to reflect on their experiences of learning programming but to also discuss what they continued to encounter as problematic. The aim was to see whether there was any commonality between the concepts proposed from the three different sample populations and put them under further scrutiny as candidates for Threshold Concepts.

Key Findings

The focus group transcripts were thematically coded to look for evidence of Threshold Concepts as described above. Concepts which were mentioned in more than three of the five focus groups were analysed for evidence of being both troublesome and transformative. Concepts which were identified by all participants as having just one of the characteristics were discarded. The results can be found on the following page.

Programming Concept	Dimension of Threshold Concept Theory	Data Examples
Classes and inheritance	*Troublesome* and *transformative* by practitioners, *troublesome* and *transformative* by students	"Classes and inheritance for me… yes, getting your head around that was quite hard but then once you, it becomes quite a vital part of the programing once you get your head around it" *year 1 student participant*
Designing objects	*Troublesome* by practitioners, *troublesome* and *transformative* by students	"When I properly understood how to separate everything out into appropriate classes that was like a major turning point in probably my ability as a programmer" *year 1 student participant*
User Interface Architectures	*Transformative* by practitioners, *troublesome* and *transformative* by students	"I feel like User Interface Architecture just grows bigger and bigger at some point and you have to be organised not to lose any piece of code and to actually be able to find, like... adding, like, two lines of code" *year 1 student participant*
Data structures	*Troublesome* by practitioners, *troublesome* and *transformative* by students	"(Data Structures) weren't hard to understand, but we don't, we all understand, we should understand it a little bit better than what we currently do. And definitely the first year we felt that way" *year 3 student participant*
Regular expressions	*Troublesome* by practitioners, *troublesome* and *transformative* by students	"You just look up a regular expression, no one ever; I don't think anybody in the universe writes... You can look it up, you can sit with a book and work it out like any mathematical thing, and then it's done, and you look at what you've worked out and you don't understand it. It doesn't matter, it works." *Practitioner participant*
Abstract classes	*Transformative* by practitioners, *troublesome* by students	"And abstract classes for me anyway, I don't know about anyone else… Yes, I mean I understand it I just don't know how like I just I've never seen the point in using it" *year 1 student participant*

Threshold Skills

As part of the lab observations one of the student Teaching Assistants also suggested that the nature of programming is skill-based, one that only make sense in application, and so we might consider the place of skills in the study. A similar notion has been put forward in some of the Threshold literature (Thomas et al., 2014). One possible Threshold Skill was identified throughout the course of the study; **Code Organisation** was considered to be both troublesome and transformative by all of the focus groups:

"Organisational code… is definitely one of the... it takes experience to write organised code." (practitioner participant)

"I think you learn about code organisation throughout your life." (year 3 student participant)

"you have to be organised not to lose any piece of code and to actually be able to find, like... adding, like, two lines of code." (year 1 student participant)

Emotional response

Emotional responses frequently occurred or were referred to when a participant discussed a potential Threshold Concept, adding additional validity to claims of their legitimacy as such. Furthermore, it was perhaps unsurprising to find that they appeared at other times in the data. As has already been discussed, programming is difficult to learn and, it could be argued, practice. As illustrated by the following quote from one of the practitioners, the challenges of programming continue well into a professional career and when a program is successful it can elicit powerful emotional reactions:

"I don't know when I started thinking I'm a proper programmer but for me when my code goes through a code review successfully, a peer review successfully and sits in production without breaking… for say three, four weeks without any problem at all, then I start feeling proud, not proud, but content and satisfied with myself"

Accidental Complexities

An unexpected find in the data was the occurrence of what we termed 'Accidental Complexities'. Accidental complexities can be a concept that in itself isn't a Threshold Concept, but when taught alongside other larger (perhaps Threshold) concepts, might introduce additional difficulty for the learners. Examples of such concepts mentioned by students as troublesome during the study include Layout Managers, which are taught alongside the development of graphical user interfaces, and ActionListener, which is used by students as a proxy for event handlers – particularly where implemented using anonymous inner classes. Accidental complexities may also occur when attempting to provide the students with an additional skill (e.g. teamwork), or as a pedagogical device (e.g. pair programming as a means of enabling peer teaching). While the inclusion of such activities would generally be seen to be good practice, several of the student participants discussed the problematic nature of trying to amalgamate various different people's ideas together to find a cohesive solution to the problem at hand. This was cited as a particular issue when students were in teams of mixed experience or ability.

Discussion

The Threshold Concepts identified were done so through a process of consensus-building amongst the students and industry professionals. While many of the concepts discussed were identified as troublesome, only a few were also considered to be transformative by one or more of the participant groups. As a result, there is compelling evidence to suggest that the six concepts put forward are suitable candidates for Threshold Concepts in programming. Of particular interest are *Classes and Inheritance*, *Designing Objects* and *Abstract Classes*, all of which fall under the area of Object-Oriented programming. 'OO' has in the past been suggested in the literature as a prospective Threshold Concept in programming, but it has also been criticised as being far too large an area to be of significant use (Rountree & Rountree, 2009). Our findings contribute more specific sub-concepts of OO which may be of more help in identifying particular points in the curriculum where students may require additional support.

The researchers noted during the course of the study that the dimension of acquiring a new identity that is part of a community of practice may be limited in its capacity to identify potential Threshold Concepts. The students themselves didn't respond to prompting regarding 'feeling like a programmer' and many of the industry practitioners joked that they still didn't feel like they were a programmer, even after years of professional experience. The closest suggestions they made were related to skills rather than concepts, such as the ability to devise the program themselves from start to finish, as the following quote illustrates:

"A lot of people don't feel that confident in their ability, and so for me it doesn't feel like there was one concept that I grasped that necessarily made me feel like a programmer. It's more as you get more exposure to your colleagues and stuff, and I guess you're getting less criticism, you start to feel a little more confident"

The above quote reveals the range of emotional strings tied up with programming. The participant alludes to the peculiarity in programming of work product being judged largely for its ability to not go wrong rather than because of inherently good design or execution. The students similarly refer to a sense of relief that a program simply functions as required, going some way to explain the prevalence of partial conceptual understanding amongst the participants as revealed when applying Eckerdal et al.'s (2007) framework to the focus group data.

Accidental complexities were an interesting finding in the focus group data and prompts debate regarding a tension between accepted good teaching practice and Threshold Concepts and Skills. The complexities may have been introduced as a result of an attempt to provide students with additional tools and skills which, while in themselves are undeniably useful e.g. learning to work as a team or using anonymous inner classes when writing ActionListeners, may actually cause additional difficulties in grasping the main concept at hand. For instance, the association with (suggested as troublesome) ActionListener and an identified Threshold Concept brings into question the appropriateness of trying to cover both at the same time. While the smaller concepts in question may be considered essential knowledge for a proficient programmer, there is a strong argument to delay teaching them until the main Threshold Concept has been mastered. Further exploration of the satellite concepts taught alongside the identified

Threshold Concepts may reveal additional opportunities to strip back and simplify the curriculum to spend more time on the concepts which take priority and ensure the student has successfully traversed their so-called ‘liminal space’. Additionally, there is an argument for a reconsideration of which points in the curriculum are suitable to be taught as group work, as their coinciding with a Threshold Concept may prove problematic. While there may be strong justification for introducing this complexity at this point, the level of support provided or the scale of the complexity introduced (e.g., team size, level of independence of the teams, etc.) can negatively impact on students passing through liminal space.

The findings of the study have produced some interesting questions. Now that we have identified some potential Threshold Concepts, which is the best way to approach structuring the syllabus around them? Is spacing them out across the terms of the first year enough? Does the order of the concepts being taught make a difference to how challenging they are? Furthermore, how can we best decide what is essential to be taught alongside Threshold Concepts in order to find the balance between including what is necessary at that particular time and what is just unhelpful noise? These are issues we intend to explore for the next steps of the study.

Acknowledgements

This research took place with invaluable input from undergraduate students and student Teaching Assistants within the King’s College London Informatics Department as well as programmers from industry. In particular, our special thanks go to the first year undergraduate students and the industry practitioners who volunteered their time to participate in the study.

References

Baillie, C., Bowden, J.A. & Meyer, J.H.F. (2012) Threshold capabilities: threshold concepts and knowledge capability linked through variation theory. *Higher Education*, 65(2): 227–246.
Barradell, S. (2012) The identification of threshold concepts: a review of theoretical complexities and methodological challenges. *Higher Education*, 65(2): 265–276.
Davies, P. (2006) Threshold concepts: how shall we recognise them? In *Threshold Concepts and troublesome knowledge*, Edited by: Meyer, J. H. F. and Land, R. London: Routledge.
Eckerdal, A., McCartney, R., Moström, J.E., Ratcliffe, M., Sanders, K. & Zander, C. (2006) *Putting threshold concepts into context in computer science education*, ACM.
Eckerdal, A., McCartney, R. & Moström, J.E. (2007) *From Limen to Lumen: computing students in liminal spaces*, New York, New York, USA: ACM.
Hughes, J. & Peiris, D. (2006) *ASSISTing CS1 students to learn: learning approaches and object-oriented programming*, ACM.
Jenkins, T. & Davy, J. (2015) Diversity and Motivation in Introductory Programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, 1(1): 1-9.
Lahtinen, E., Ala-Mutka, K. & Järvinen, H.-M. (2005) *A study of the difficulties of novice programmers*. In ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education. ACM, 14–18.
McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4): 125–180.

Meyer, J. H.F. & Land, R. (2003) Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher Education*, 49(3): 725–734.
Pedroni, M., Meyer, B. & Oriol, M. (2009) What Do Beginning CS Majors know? *Technical Report 631*, ETH Zürich, Chair of Software Engineering.
Robins, A., Rountree, J. & Rountree, N. (2003) Learning and teaching programming: A review and discussion. *Computer science education*, 13(2): 137–172.
Rountree, J. and Rountree, N. (2009) *Issues regarding threshold concepts in computer science*. Proceedings of the eleventh Australasian computing education conference (ACE 2009). Edited by: Hamilton, M. and Clear, T. pp.139–145. Darlinghurst, Australia: Australian Computer Society.
Thomas, L., Boustedt, J. & Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K. & Zander, C. (2014) *A broader threshold: Including skills as well as concepts in computing education*. Fourth Biennial Conference on Threshold Concepts: From personal practice to communities of practice, Trinity College, Dublin.
Zander, C., Boustedt, J., Eckerdal, A., & McCartney, R. (2008) Threshold concepts in computer science: A multi-national empirical investigation. In *Threshold Concepts Within the Disciplines*, Edited by: Land, R. Meyer, J. H. F. and Smith, J. Rotterdam: Sense.