Issues Regarding Threshold Concepts in Computer Science

Janet Rountree **Nathan Rountree**

Department of Computer Science,
University of Otago,
Dunedin, New Zealand
Email: {janet,nathan}@cs.otago.ac.nz

Abstract

Threshold Concepts deserve discussion and reflection in Computer Science Education; they provide a conceptual framework intended to re-empower tertiary educators. At this stage, the idea of Threshold Concepts raises plenty of questions, promises renewed learner and teacher engagement, and suggests a means of focusing on the key aspects of a discipline that will allow a learner to, for example, "think more like a computer scientist." But what precisely are threshold concepts? Can we identify them? Can we agree on which concepts are threshold concepts and which are not? Can we validate them? If threshold concepts do exist, and can be identified and agreed upon, then how would they alter what we teach, how we teach, and how we assess? Do threshold concepts represent anything new or unexpected? The purpose of this paper is to set out issues for the Threshold Concepts model in Computer Science Education and encourage on-going discussion.

Keywords: Threshold Concepts, Computer Science Education, Liminal Space

1 The Notion of Threshold Concepts

The Threshold Concepts model is fashionable. The notion has rapidly gained popularity since being first proposed in 2003 (Meyer and Land, 2003), with the second Threshold Concepts Conference recently being held at Queen's University in Canada (`http://thresholdconcepts.appsci.queensu.ca`), two books in print (Meyer and Land, 2006a; Land et al., 2008), as well as the publication of many topical articles across a variety of disciplines. The idea has struck a chord with many academics interested in research into the teaching of their discipline and its practice. Examples include Biology (Taylor, 2006), Economics (Davies and Mangan, 2007; Shanahan and Meyer, 2006), Accounting (Lucas and Mladenovic, 2007), Electrical Engineering (Carstensen and Bernhard, 2008), Statistics (Dunne et al., 2003), Geology (Stokes et al., 2007), and Marketing (Lye, 2006). In Computer Science Education (CSE) there is also a developing context for Threshold Concepts (Eckerdal et al., 2006). The purpose of this paper is to discuss work done on Threshold Concepts in CSE, to consider notable issues, and reflect on the usefulness of this conceptual framework to our discipline. For example, we should ask ourselves whether it is possible to agree upon which concepts are threshold concepts and which are not. Can threshold concepts be validated, and how would they alter what we teach, how we teach, and how we assess? We should also consider whether they represent anything new or unexpected.

We can view Threshold Concepts in two parts: first, as a model or framework, and second, as "instance" examples. (To distinguish between instances of threshold concepts, and Threshold Concepts as a model, we shall capitalise the latter.) In the first case, Threshold Concepts provides a model for academics in higher education to develop their teaching and support student learning. The conceptual framework is intended to re-situate teaching and learning within the context of its own discipline, in contrast to the role *learning outcomes* have developed as a managerial tool to audit and monitor "success" (Hussey and Smith, 2003). To contrast the two models, learning outcomes treat education as a set of activities designed to achieve a set of pre-specified outcomes, with success defined in terms of meeting those outcomes. Typically, the outcomes are phrased, "by the end of the course the learner will be able to. . . ." Threshold concepts, on the other hand, state that learners go through a *transformation*, after which they begin to "think more like a computer scientist," and that they gradually acquire the identity of a community of practice. During this transformation, certain parts of the curriculum are pivotal: they represent the "portals" that learners must traverse in order to succeed. To be considered a member of the community of practice, mastery of these concepts is *required*, and the process of mastery is seen as a sort of rite of passage.

Secondly, the term "threshold concept" is used to refer to any part of the curriculum that should be treated as one of these portals. They may be recognised by (probably) being all five of: transformative; irreversible, integrative, bounded, and troublesome (Meyer and Land, 2006b, p7–8). Threshold concepts are transformative in nature because their comprehension creates in the learner a new way of viewing and describing the subject and may alter the learners' perception of themselves and the world. Threshold concepts are irreversible since the fundamental qualitative change that occurs is unlikely to be unlearnt. It follows that threshold concepts are also integrative because learners make new connections, perceive previously unknown relationships, and accordingly change their sense of the world. They are described as bounded, or as boundary makers (Eckerdal et al., 2006, p103), since a threshold concept "...helps to define the boundaries of a subject area because it clarifies the scope of a subject community" (Davies, 2006, p74). The final characteristic is that threshold concepts embody knowledge that is troublesome for learners to grasp—it is more than simply new subject matter, it is material that is diffi-

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 95, Margaret Hamilton and Tony Clear, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

cult and possibly counter-intuitive, and accordingly it cannot, as yet, integrate with the learner's current mental schema.

The two most important of the five characteristics of threshold concepts are the troublesome and transformative nature of the knowledge. Troublesome knowledge goes beyond knowledge that is difficult to understand—it is tied up with incorrect or incomplete mental models, misconceptions, the inability to transfer understanding from one context to another, conflicts with current understanding or perspective, emotional response, (e.g. being vexing), and tacit presumptions (Perkins, 2006). Working through problems and gaining understanding of troublesome aspects of the subject matter is interpretative: it extends the use of language; modifies ways of thinking and practising; it is a process of "identity formation" through which the learner gains entrance into the subject community (Davies, 2006). The Threshold Concepts framework views "... learning as a form of journey, during which the student not only gains insights great and small, but is also changed as an individual by new knowledge" (Meyer and Land, 2007). In our case, the learner begins to think and practice 'more like a computer scientist.'

All threshold concepts are also core concepts, but not all core concepts are threshold concepts. For example, the ACM "Curriculum Guidelines for Undergraduate Degree Programs in Computer Science" presents a body of knowledge including core topics. Picking topics arbitrarily, "object-oriented programming" may be a threshold concept, but the "history of computing," whilst recognised as a core topic, may not be a threshold concept. The "history of computing" (in this argument) might be new knowledge, it might be complex knowledge, it might require effort to learn, but it is not troublesome or transformative in nature for the learner.

Let us take recursion as a potential example of a threshold concept in CS: does recursion meet the five defining characteristics? We expect that many CS educators would agree that recursion presents an example of troublesome knowledge. When first seen the notion of 'self reference' is alien and many novice programmers struggle to come to terms with this concept in implementation and description. Once a learner 'gets' recursion she has a significant transformation in her mental process. For example, she sees recursive sets rather than the state a program gets into, ways of describing program state change to include base/stopping cases related to the atoms of a recursive set. Elegance in program design is brought into focus. She starts to make connections and see relationships with other material such as the fact that all loops can be expressed as recursion. It is unlikely that this new understanding will be unlearnt, so it follows that recursion is irreversible as well as being integrative. On the one hand recursion is useful in the practice of programming, on the other hand it is a theoretical construct that defines what is computable; hence recursion is a boundary marker for both Software Engineering and Theoretical Computer Science.

2 Liminal Space and Pre-liminal Variation

One of the most powerful inferences that can be drawn from Threshold Concepts is that of *liminal space*. Meyer and Land suggest that a threshold concept is rarely mastered in a single "aha" moment, but instead requires a period of time over which a student makes the transition. The period of transition is referred to as "liminal space" (from the Latin *limen*, meaning boundary or threshold). Students who are in the period of transition may be characterised as undergoing a "rite of passage," at the end of which they will have achieved new knowledge and status within their community. The rites may be drawn-out, confusing, and require that students begin to think and act differently to be seen as having succeeded. (Success here is defined as not "understanding how a practitioner thinks," but "beginning to think like a practitioner.")

The proponents of Threshold Concepts characterise liminal spaces as the places where students "get stuck" if they are going to get stuck at all. Students show such a range of ability to traverse the liminal space that it is natural to think of them as being "effective" or "ineffective" at negotiating the liminality. Meyer and Land suggest therefore that *pre-liminal variation* is the key to understanding how and why students might effectively negotiate liminal space. What is there in each student's background that might help or hinder their liminal journey? In examining pre-liminal variation, we may need to attend to more than just whether or not they have mastered the academic material considered to be pre-requisite; perhaps also we need to examine their epistemological stance: are they ready to build knowledge in the way we expect them to, and will they be able to tolerate the (possibly quite long) period of uncertainty, confusion, and even oscillation between seeming to have "got it," and feeling sure that "it" will remain forever elusive?

Eckerdal et al. (2007) present research to support the notion that students in Computer Science can be accurately characterised as spending time in liminal space. Interviews with students regarding things that they had previously identified as potential threshold concepts established that all of the proposed features of liminal spaces were evident: significant time commitment, oscillation between states, emotional involvement of anticipation and anxiety, and mimicry of the new state.

The implications attendant upon *liminal space* may well prove as significant as the Threshold Concepts model itself. First, Eckerdal et al. note that, contrary to popular conceptions of "levels" of understanding, students passing through liminal space cope with different aspects of concepts (theoretical, practical, etc.) in parallel. Second, that the *time* required to make the transitions is significant, and perhaps unexpected to novice students (and perhaps to those setting learning outcomes). Third, that there is a significant *emotional* reaction to dealing with liminality, and that such reaction is normal and should be managed rather than ignored or dismissed. Finally, that mimicry during the negotiation of liminal space may not be undesirable, but may be a normal part of the process of coming to terms with conceptual difficulty.

3 Identification of Threshold Concepts

Assuming that the Threshold Concepts model is valid, what processes can we use to identify threshold concept instances? Davies (2006) notes that threshold concepts provide a method of describing the 'way of thinking' distinctive to a discipline; a method that is an alternative to the 'key concepts' idea or the method of phenomenography. However, he also notes that identification of threshold concepts may be difficult due to their being "taken for granted" within a subject, and "therefore rarely made explicit." He goes on to suggest two methods for recognising the threshold concepts within a discipline. (For convenience we shall refer to them as the *first approach* and the *second approach*.) The first approach argues that we might recognise threshold concepts by examining the different ways in which two disciplines

analyse the same situation. For instance, if Social Scientists analyse school choice as a zero-sum game, but Economists as a problem of general equilibrium, that might lead us to believe that equilibria represent a threshold concept in Economics.

The second approach to identifying threshold concepts described by Davies is to focus on the distinction between people inside and outside the community of practice—specifically, on the differing ways in which students and experts in the field analyse the same problem, or group of problems. This, of course, is empirically very convenient for educators in a given field, as they have the best opportunities to conduct research on their own students. Consequently, most work on identifying threshold concepts within disciplines has focused on this approach. The advantage is that it allows researchers to look at problems that only exist within one field. The clear disadvantage is that there is no equivalence between novice/expert comparisons and expert/expert comparisons.

Most substantial work on identifying threshold concepts in Computer Science has taken the *second* approach, examining the responses of *students* in Computer Science to questions about where they got "stuck" while studying. (See the comments on studies in Computer Science in Section 5.) To date, the first approach (examining the differences in methodology between related fields) has been largely ignored. This seems a missed opportunity, since Computer Science has a wealth of related fields, many of which have shared interests. Examining the different ways in which practitioners in Computer Science, Information Systems, Mathematics, Physics, Electrical Engineering, and Linguistics, tackle similar problems may produce excellent candidates for threshold concepts in each discipline, and opens up a research question concerning whether threshold concepts are shared between disciplines (and thus whether there is a hierarchy of threshold concepts), and whether threshold concepts mutate as they cross between disciplines.

Work has already begun on validating the Threshold Concepts model and on identifying instances of threshold concepts in Computer Science, e.g. in Eckherdal et al.'s multi-national study. This research has made it clear that students in Computer Science encounter things that look much like threshold concepts and liminal spaces, but that there are difficulties in articulating the *granularity* of such things. For instance, both lecturers and students referred to "object-orientation" as a threshold concept, but the authors note that this is almost certainly too broad a term, when interviews reflected that the "stuck places" were more at the level of polymorphism or object cooperation. Work on curriculum design by Mead et al. (2006) may provide a way of teasing apart these hierarchical distinctions; they propose the idea of an *anchor concept* (which is a concept that is either foundational or transformative AND integrative) and an associated anchor concept graph, which maps the cognitive load shared by related anchor concepts. This approach might allow us to specify concepts at multiple levels of granularity, and in a logical order that recognises the dependency of threshold concepts on other foundational material.

Some logical difficulties in identifying threshold concepts have been identified. Rowbottom (2007) raises several general caveats, all of which apply to threshold concepts in Computer Science (and, indeed, any other subject). His notes of caution are as follows:

1. The features attributed to threshold concepts are insufficiently precise to distinguish them from any other concept. They are described by their originators as *probably* and *not necessarily* transformative, irreversible, troublesome, etc. Thus, any concept you care to mention might be a threshold concept, even though it has none of the features, and any concept that has all of the features may not in fact be a threshold concept. Thus, I may argue that "scope" is a threshold concept in Computer Science, and you may argue that it is not, but neither of us can properly appeal to the definitions to support our argument. Without those definitions, it is not logically feasible even to use empirical research to support or refute a claim that something is or is not a threshold concept.

2. There are at least three accounts of what constitutes a *concept*, from Cognitive Science (mental models functionally equivalent to symbols or words, complete with combinatorial syntax and semantics), to competing views in Philosophy (the concept of X is reducible to the *ability* to think of Xs or classify things as Xs; or concepts as abstract entities of thought associated with names). Possibly these views are not contradictory, and possibly we are meant to default to the view of Cognitive Science. But which view we hold will profoundly affect our method of determining whether a particular concept has been mastered. Rowbottom presents "playing tennis" as an example of an activity where there is a distinction between *knowing that* and *knowing how*; we might just as readily suggest that programming is an activity where changes in concept do not necessarily result in changes of practice, nor that changes in practice are guaranteed to be a result of a change in concept.

3. Not only are the qualifiers attached to each feature a problem, but so too are the features themselves. Take "transformative" as an example. What may be transformative for me, may not be transformative for you. For instance, if I learned Pascal as my first programming language, then the notion of *generics* would tend to have all the features of a threshold concept—in particular, troublesome, transformative, and integrative—because Pascal does not have the features necessary to support truly generic container types (specifically, untyped pointers, or the ability to specify that all the objects in the container will be the same size if not the same type). [1] In contrast, a student who has Java as a first language is likely to have no difficulty at all with the idea that a container can store things of many types, since containers can always be defined as storing things of type "Object." Thus, *genericity* is unlikely to have any of the connotations of a threshold concept to a student who has Java as her first programming language.

These caveats are not necessarily enough to dismiss the Threshold Concepts model, nor the possibility of identifying good candidates for threshold concept instances. Even though our definitions of threshold concepts may not be perfectly precise, we can defeasibly posit their existence, and agree upon their most distinctive features, until such time as we find evidence to suggest that we should retract our assertion. Imprecise definitions are insufficient evidence for retraction; "four-legged mammals" might be an imprecise definition for "cats", but that does not imply that cats do not exist. However, these problems

[1] Variant records will allow a programmer to contain a set of specified types, but will not allow the containment of any new type without modification of the record. Furthermore, if one record holds things of type A or type B, code that processes the record has to deal with the possibility that type A processing **can** be invoked on type B objects.

with the identification of threshold concepts do indicate that there is only a limited amount of utility in trying to determine threshold concepts by empirical means. Science, along with its attendant methods and expectations, is a social construction: if there are threshold concepts, and they are really as critical as the model claims, it is because *we* have put them in place and made them so. Studying our students for signs of threshold concepts may help to make our tacit assumptions explicit; but studying differences in practice between experts in related fields should provide a broader set of concepts that act as boundary markers for a subject.

4 Consequences of Threshold Concepts

Land et al. (2006) state that the Threshold Concepts idea "presents important challenges for curriculum design and for learning and teaching." Specifically, they draw attention to nine considerations that they feel are important with respect to the design and evaluation of curricula. These are:

1. that threshold concepts are the "*jewels in the curricula*," demanding longer and sharper focus than other concepts;
2. that they emphasise the *"importance of engagement"* by stressing the transformation of the student into someone who thinks like a computer scientist, rather than someone who understands how computer scientists think;
3. that they emphasise *listening for understanding* on the part of the teacher—in particular, listening for the signs of pre-liminal variation among students;
4. that they emphasise the *reconstitution of self* on the part of the student, and imply the design of an environment supportive to the discomfort of repositioning oneself in relation to the subject;
5. that they provide good reasons for *tolerating uncertainty* both on the part of students and teachers, due to the time it takes to negotiate the liminal space of a Threshold Concept;
6. that they promote *recursiveness and excursiveness* of learning—that troublesome knowledge often requires re-visiting, and that the "outcome" of learning is not just a set of "the learner will be able to…" statements, but that the learner will have been transformed by the journey into one who thinks differently;
7. that understanding *pre-liminal variation* among students will help us to understand why some students negotiate the curriculum effectively, while others have more difficulty;
8. that they may expose the *unintended consequences of generic 'good pedagogy,'* in that they provide examples where standard methods (such as simplifying the concept to begin with) prove dysfunctional;
9. that they highlight an aspect of the *underlying game*—that is, that students will only become members of the community of practice if they master the *authorised* understanding of threshold concepts, and that alternative versions (based on personal experience or common-sense) will place them in unwitting opposition to the community.

At first glance, it is easy to dismiss these considerations as "nothing new here." After all, in Computer Science, we are only too well aware that some topics need longer and greater emphasis than others, that student engagement is paramount to success, that student variation is immense and requires significant adaptability on the part of teachers. We know all of these things, and using the language of Threshold Concepts to express them is unlikely to affect our understanding of them, nor provoke a radical shift in our strategies for dealing with them.

However, there remains the possibility that the Threshold Concepts model may provide *unexpected* consequences. For instance, we can reason as follows: threshold concepts are integrative; they allow the practitioner to combine other fundamental concepts in ways unique to the discipline. If we believe object-orientation to be a threshold concept, what other fundamental concepts are being integrated? A strong implication of the Threshold Concepts model is that those concepts that are to be integrated should be grasped first by learners, before attempting to traverse the liminal space where they will learn to integrate them. If, for instance, we view the encapsulation of state and behaviour as a key part of object-orientation, the implication is that state and behaviour should be mastered first. This suggests that an objects-first approach to CS1 is incompatible with the Threshold Concepts model. To date, there has been little or no work on exploring the consequences of Threshold Concepts in this fashion: what teaching practices would we adopt, and what should we reject if the Threshold Concepts model is valid?

At a higher level of abstraction, is it possible—or even necessary—that we can ever come to a general agreement on what constitute threshold concepts in Computer Science? It could be argued both ways: that we can and should, and that we cannot and that it does not matter. For the latter argument, we need only point to the ACM curriculum and state that some topics will be troublesome for some learners, and different topics will be troublesome for other learners. We adapt as necessary, listening carefully to our classes for signs of difficulty and for indications of dawning mastery. In adopting that attitude, the Threshold Concepts model tells us nothing new, leads to no insights, and has no implications. On the other hand, what happens to an academic subject when the *underlying game* (a phrase used often in Threshold Concepts literature) remains implicit, and is never made explicit? Even in Economics, where different schools of thought (such as the Austrian School, or Keynsian Economics) provide very different analyses of the same events, they all agree that they are *doing economics*, based on a shared set of underlying, unifying concepts.

Those concepts *can* be identified by focusing on what we do that is peculiar to our discipline, and by making explicit those things that we think everybody agrees on. Some of that identification can no doubt be achieved by observing students making the transition from not thinking like a practitioner to doing so, and some can be achieved by observing how practitioners in different but related fields practice differently. It seems reasonable that the two different methods will highlight different concepts.

5 Studies in Computer Science

Shinners-Kennedy (2008) proposed *state* as a threshold concept in Computer Science, showing that it meets all five threshold concept criteria. Vagianou (2006) considers *program-memory interaction* as a possible example of a threshold concept in Computer

Science. The author argues that research shows a viable computer model must be present before programming is engaged, and that an introductory programming course needs to shift each student's viewpoint from that of a non-expert, "end-user" stance toward a "programmer stance" with an awareness of being directly responsible for the computing process. Discussion suggests that Program/memory interaction displays the characteristics of a threshold concept. The notion is troublesome, since beginning students do not realise *how* the use of memory takes place, or their role in that process. It acts as a boundary marker because our biological concept of short and long term memory differs from computing memory. It is integrative, showing otherwise hidden relationships between hardware and software. Program/memory interaction is transformative, since once understood it will significantly shift the student's perspective and it follows that the notion should be irreversible since the knowledge is very unlikely to be unlearnt.

A paper by Khalife (2006) aims to identify potential Threshold Concepts in introductory programming courses and propose solutions to help students surpass thresholds. The author presents some commonly accepted novice programmer difficulties (such as lack of problem solving strategies), and then suggests that the first threshold a student needs to pass is "... to develop a simple but yet concrete mental model of the computer internals and how it operates during program execution" (Khalife, 2006, p246). A computer model for teaching purposes is then set out along with results of an empirical evaluation of that model.

To date, the most systematic and in-depth approach to studying Threshold Concepts in Computing is an on-going multi-institutional, multi-national series of projects underway in the UK, USA, and Sweden (Zander et al., 2008; Eckerdal et al., 2006; Boustedt et al., 2007; Eckerdal et al., 2007; Moström et al., 2008). Discussions by members of this group have been underway since (at least) early 2005 when, at the Conference on Innovation and Technology in Computer Science Education in Portugal, they "... interviewed 36 Computer Science educators from nine countries and asked for suggestions about concepts that met the criteria for a threshold concept" (Zander et al., 2008). There was no universal consensus of concepts amongst academics, but the most popular are listed as: levels of abstraction; pointers; the distinction between classes, objects, and instances; recursion and induction; procedural abstraction; and polymorphism. At the 5th Koli Calling conference in Finland, McCartney and Sanders (2005) presented a poster on Threshold Concepts in CSE and collected opinions from delegates on potential Threshold Concepts through questionnaires and interviews. At ITiCSE'06, the authors provided a helpful discussion of related areas of research in CSE, namely: constructivism; mental models; misconceptions; breadth-first approach to teaching; and, ideas fundamental to the discipline (Eckerdal et al., 2006). The paper continues by proposing, with support from literature, two candidates for threshold concepts—abstraction and object-orientation. A paper presented at *SIGCSE'07* answers *yes* and *yes* to the title "Threshold Concepts in Computer Science: Do they exist and are they useful?" (Boustedt et al., 2007). The principal contribution of this paper is to describe their empirical approach of using structured interview techniques to identify candidate threshold concepts in Computer Science. From 33 concepts mentioned by students and educators, they examine two in detail to establish that they have the required features: object-orientation and pointers. Here, they make the point that what they might have uncovered are "... perhaps broad areas in which thresholds exist." Interview subjects were inclined to use the broad terms to identify the concepts, but then spoke in much more specific terms about problems they encountered within those areas.

What the on-going multi-national study provides is validation of the Threshold Concepts model for CSE. Possible "instance" examples of threshold concepts (which display the five threshold criteria), have been identified by practitioners from different countries. CS students were interviewed regarding topics they had found troublesome and "got stuck" on, and an intersection of topics identified by both (object-oriented programming and pointers) have been investigated in greater detail for evidence that they satisfy the threshold concept criteria. The multi-national study has also looked at the idea of liminal space as an appropriate description of the transitional space CS students negotiate as they develop ways of thinking and practising.

6 Concluding Comments

The Threshold Concept Model presents a disciplinary situated learning framework for higher education which is a welcome shift in perspective away from the checkbox flavour of learning outcomes. "Instance" examples of threshold concepts are core curriculum concepts with the particular properties of being transformative, irreversible, integrative, boundary markers, and troublesome. In Computer Science, proposed threshold concept examples include: state; program-memory interaction; levels of abstraction; pointers; the distinction between classes, objects, and instances; recursion and induction; procedural abstraction; and polymorphism. With further work, is it likely that academics in CSE would agree on a set of threshold concept topics? Perhaps! We think it likely that there will be agreement for some threshold concepts and not others. In the empirical sense, it is not possible to validate threshold concepts because what is a threshold concept for one person, may not be for another. The best we can achieve is a sense of "many" or "most" learners finding a particular topic meets the requirements of a threshold. Does this lack of validation matter for CSE? No, we think not, because practitioners define the subject; we define the curriculum (not the students), and so empirical validation using students may be something of a "red herring." If threshold concepts define the boundaries of how practitioners perceive a subject, then surely we need to study practitioners if we wish to define the threshold concepts.

If practitioners have differing perspectives then you simply get different schools of thought. Will threshold concepts alter what and how we teach and how we assess Computer Science? Yes and No. On one hand, for example, object-orientation is troublesome and counter-intuitive (we already know this and give it extra teaching emphasis) so labelling it a threshold concept provides no additional enlightenment. On the other hand, investigating students' *liminal space* as they come to terms with object-orientation will provide valuable insight into what makes an effective novice. Knowing what makes a novice effective in traversing liminal space allows particular skills and ways of thinking to be targeted. Discovering an unexpected threshold concept would be of much interest. The implications of threshold concepts are also of interest for CSE; for example, is an objects-first approach in teaching incompatible with Threshold Concept theory? Threshold Concepts literature strongly suggests that the integrative nature of thresholds requires students to first have mastered some fundamental concepts before embarking on the

threshold concept itself. Simplification of a threshold concept to make it easier has been shown (in Economics at least) to lead students to settle for a naive version of that knowledge (Land et al., 2006, p333). An implication for CSE could be seen as avoiding teaching "objects-first" in CS1.

From a philosophical viewpoint the difficulty with Threshold Concepts is the lack of a formal definition which would allow specification of what is, and what is not a threshold concept. The model needs to become more precise because, as it stands, no threshold concept candidate can be verified—it is only possible to make an assertion that it exists. We suggest that one small step to help clarify the model is to view Threshold Concepts in two parts: first, as a model or framework, and second, as "instance" examples. Both need to be validated separately—are there subjects for which the Threshold Concepts model is completely invalid? Or does it work for every tertiary subject? If you have a subject for which Threshold Concepts is a valid model, what are the threshold concepts within it? How can you tell for sure? How can you tell if you have identified all of them? Is it possible to distinguish those things that look like threshold concepts but are not?

Although we have titled this paper "*Issues with Threshold Concepts in Computer Science*," and have noted some difficulties, we do not think that the notion of Threshold Concepts should be dismissed. As educators in Computer Science there are essential questions which we continually ask: When do we consider a student has been successful? Why are some students successful, whilst others are not? And how do we support students through their learning process? In relation to these questions, there are three aspects of particular value that Threshold Concepts contribute. Threshold Concepts provide:

1. an apt description for what it means for a student to have been successful in our discipline;
2. the addition of epistemological concepts of *liminal* and *pre-liminal space*, which gives direction for future research into what makes an effective novice programmer;
3. a focus on our *community of practice*, giving deference to the disciplinary knowledge of the academic, so that concerns are "... always analysed and resolved from, and within, specific and situated disciplinary contexts" (Meyer and Land 2007, p14).

Where should the Threshold Concept discussion for CS education go next? We suggest that Davies' *first* approach to identifying threshold concepts—examining the ways in which practitioners in related disciplines solve similar problems—provides the best avenue for further research. If the goal is to identify "how to think like a Computer Scientist" then we must first study the practitioners, not their students. In CS the first challenge is to specify what constitutes our subject. If we ask our colleagues what it means to be a Computer Scientist, how much agreement will there be? For example, how much overlap will there be between groups with a software engineering flavour, or theoretical, or electrical engineering approach? The immediate value of Threshold Concepts in CS Education is to require us to address what it means to be a Computer Scientist.

References

Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., and Zander, C. (2007). Threshold concepts in computer science: do they exist and are they useful? *SIGCSE Bull.*, 39(1):504–508.

Carstensen, A.-K. and Bernhard, J. (2008). *Threshold Concepts and Keys to the Portal of Understanding: Some Examples from Electrical Engineering*, pages 143–154. In (Land et al., 2008).

Davies, P. (2006). Threshold concepts: How can we recognise them? In (Meyer and Land, 2006a), pages 70–84.

Davies, P. and Mangan, J. (2007). Threshold concepts and the integration of understanding in economics. *Studies in Higher Education*, 32(6):711–726.

Dunne, T., Low, T., and Ardington, C. (2003). Exploring threshold concepts in basic statistics, using the internet. In *AISE/ISI Satellite*, Berlin.

Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., and Zander, C. (2006). Putting threshold concepts into context in computer science education. *SIGCSE Bull.*, 38(3):103–107.

Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., Thomas, L., and Zander, C. (2007). From limen to lumen: Computing students in liminal spaces. In *ICER '07: Proceedings of the Third International Workshop on Computing Education Research*, pages 123–132, New York.

Hussey, T. and Smith, P. (2003). The uses of learning outcomes. *Teaching in Higher Education*, 8(3):357–368.

Khalife, J. T. (2006). Threshold for the introduction of programming: Providing learners with a simple computer model. In Romero, P., Good, J., Acosta, E., and Bryant, S., editors, *Proceedings of the 18th Workshop of the Psychology of Programming Interest Group*, pages 244–254.

Land, R., Cousin, G., Meyer, J. H., and Davies, P. (2006). Implications of threshold concepts for course design and evaluation. In (Meyer and Land, 2006a), pages 195–206.

Land, R., Meyer, J. H., and Smith, J., editors (2008). *Threshold Concepts Within the Disciplines*. Sense Publishers.

Lucas, U. and Mladenovic, R. (2007). The potential of threshold concepts: An emerging framework for educational research and practice. *London Review of Education*, 5(3):237–248.

Lye, A. (2006). Threshold concepts: Reflections on marketing education. In *Proceedings of the 2006 ANZMAC Conference*, Brisbane.

McCartney, R. and Sanders, K. (2005). What are the "threshold concepts" in computer science? In *Proceedings of the 5th Baltic Sea Conference on Computing Education Research (Koli Calling 2005)*, page 185.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., Clair, C. S., and Thomas, L. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *SIGCSE Bull.*, 38(4):182–194.

Meyer, J. H. and Land, R. (2003). Threshold concepts and troublesome knowledge – linkages to ways of thinking and practising. In Rust, C., editor, *Improving Student Learning Theory and Practice – Ten Years On*, pages 412–424. OCSLD, Oxford.

Meyer, J. H. and Land, R., editors (2006a). *Overcoming Barriers to Student Understanding: Threshold Concepts and Troublesome Knowledge*. Routledge.

Meyer, J. H. and Land, R. (2006b). Threshold concepts and troublesome knowledge: An introduction. In (Meyer and Land, 2006a), pages 3–18.

Meyer, J. H. and Land, R. (2007). Stop the conveyer belt, I want to get off. *Times Higher Education Supplement*, page 14. Issue 1807, 17 August 2007.

Moström, J. E., Boustedt, J., Eckerdal, A., McCartney, R., kate Sanders, Thomas, L., and Zander, C. (2008). A multi-national, multi-institutional project on threshold concepts in computer science: Results and implications. In *Threshold Concepts Conference 2008*, Queen's University, Kingston, Canada.

Perkins, D. (2006). Constructivism and troublesome knowledge. In (Meyer and Land, 2006a), pages 33–47.

Rowbottom, D. P. (2007). Demystifying threshold concepts. *Journal of Philosophy of Education*, 41(2):263–270.

Shanahan, M. and Meyer, J. H. (2006). The troublesome nature of a threshold concepts in economics. In (Meyer and Land, 2006a), pages 100–114.

Shinners-Kennedy, D. (2008). The everydayness of threshold concepts: 'State' as an example from computer science. In (Land et al., 2008), pages 119–128.

Stokes, A., King, H., and Libarkin, J. (2007). Research in science education: Threshold concepts. *Journal of Geoscience Education*, 55(5):434–438.

Taylor, C. (2006). Threshold concepts in biology: Do they fit the definition? In (Meyer and Land, 2006a), pages 87–99.

Vagianou, E. (2006). Program working storage: A beginner's model. In Berglund, A. and Wiggberg, M., editors, *Proceedings of the 6th Baltic Sea Conference on Computing Education Research (Koli Calling 2006)*, pages 69–76.

Zander, C., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., and Sanders, K. (2008). Threshold concepts in computer science: A multi-national empirical investigation. In (Land et al., 2008), pages 105–118.