From *Limen* to *Lumen*: Computing students in liminal spaces

Anna Eckerdal
Department of Information Technology
Uppsala University
Uppsala, Sweden
Anna.Eckerdal@it.uu.se

Robert McCartney
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT USA
robert@cse.uconn.edu

Jan Erik Moström
Department of Computing Science
Umeå University
901 87 Umeå, Sweden
jem@cs.umu.se

Kate Sanders
Mathematics and Computer Science Department
Rhode Island College
Providence, RI USA
ksanders@ric.edu

Lynda Thomas
Department of Computer Science
University of Wales
Aberystwyth, Wales
ltt@aber.ac.uk

Carol Zander
Computing & Software Systems
University of Washington, Bothell
Bothell, WA USA
zander@u.washington.edu

ABSTRACT

This paper is part of an ongoing series of projects in which we are investigating "threshold concepts": concepts that, among other things, transforms the way a student looks as the discipline and are often troublesome to learn. The word "threshold" might imply that students cross the threshold in a single "aha"moment, but often they seem to take longer. Meyer and Land introduce the term "liminal space" for the transitional period between beginning to learn a concept and fully mastering it.

Based on in-depth interviews with graduating seniors, we found that the liminal space can provide a useful metaphor for the concept learning process. In addition to observing the standard features of liminal spaces, we have identified some that may be specific to computing, specifically those relating to levels of abstraction.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education—*Computer Science Education*

General Terms

Measurement, Experimentation

Keywords

threshold concepts, liminal space, learning theory

ICER '07, September 15–16, 2007, Atlanta, Georgia, USA.

1. INTRODUCTION

This paper is part of an ongoing series of projects in which we are investigating "threshold concepts": concepts that, among other things, transform the way a student looks at the discipline and are often troublesome to learn. [12] We are interested in identifying these concepts in computer science, understanding how students experience the process of learning those concepts, and designing better ways to help students with this process.

Last year, we conducted in-depth interviews of computer-science majors nearing graduation. In our initial analysis of this data, we identified some threshold concepts in computer science. [1] We also found that students report using a wide variety of strategies to make progress in learning these difficult concepts. [10]

The word "threshold" might imply that students cross the threshold in a single "aha" moment, but often they seem to take longer. Meyer and Land [13] introduce the term "liminal space," borrowed from anthropology. *Limen* is the Latin word for threshold, so this literally means "threshold space." Roughly speaking, in our context, the term refers to the transitional period between beginning to learn a concept and fully mastering it.[1] Meyer and Land's formal definition, discussed below, helped us to formulate the questions that guided our analysis of the data. Our analysis revealed several interesting aspects of how students experience this liminal space.

In Section 2, we give the theoretical background for this work. We present our research questions in Section 3, review some additional related work in Section 4, and describe our methodology in Section 5. In Section 6, we present our results: the different ways in which students experience the liminal space, as shown in our data. We discuss these results in Section 7, and close with our conclusions and future work.

[1] *Lumen* is the Latin word for the light that we hope students find when they have fully crossed the threshold.

2. THEORETICAL BACKGROUND

Meyer and Land [13] have proposed using threshold concepts as a way of characterizing particular concepts that might be used to organize the learning process. They further develop a theoretical framework, the liminal space, which specifically focuses on the process of learning such concepts.

2.1 Threshold Concepts

Threshold concepts are a subset of the core concepts within a discipline, and are characterized as being [13]:

- *transformative*: they change the way a student looks at things in the discipline.
- *integrative*: they tie together concepts in ways that were previously unknown to the student.
- *irreversible*: they are difficult for the student to unlearn.
- potentially *troublesome* (as in [15]) for students: they are conceptually difficult, alien, and/or counter-intuitive.
- often *boundary markers*: they indicate the limits of a conceptual area or the discipline itself.

The idea has the potential to help us focus on those concepts that are most likely to block students' learning. [3]

In our interviews, we have so far found evidence that *pointers* and *object-oriented programming* fulfill the criteria for threshold concepts. [1]

2.2 Liminal space

The term "liminal space" was originally used in anthropology to describe the time during which someone is passing through a rite of passage. [19] When borrowing the term, Meyer and Land list the following defining characteristics. [13] The liminal space is a space in which someone

- is being transformed
- acquires new knowledge
- acquires a new status and identity within the community

The process of being in this liminal space and crossing the threshold may

- take time, and may involve oscillation between old and new states
- involve emotions, of anticipation, but also of difficulty and anxiety
- involve mimicry of the new state

The period of adolescence, for example, has all of the characteristics of a liminal space. In adolescence, an individual is being transformed and acquiring the identity of an adult. This process takes time, involves acquiring new knowledge – how to earn a living, for example – and is often a difficult time emotionally. Adolescents, especially in the early stages, can behave like children one moment and adults the next, oscillating back and forth between the two states. And they learn to be adults, in part, by mimicking the adults around them.

In educational settings Meyer and Land emphasize the transformative character of threshold concepts as the main reason for the liminal space: "owing to their powerful transformative effects" [12, p. 10], and in [13, p. 380] the authors explain: "we see the threshold as the entrance into the transformational state of liminality."

In the learning of threshold concepts mimicry can "involve both attempts at understanding *and* troubled misunderstanding, or limited understanding, and is not merely intention to reproduce information in a given form." [13, p. 377] The authors further relate both mimicry and emotions of difficulties and anxiety to "stuck places." Identifying such stuck places in the learning process can lead to a fuller understanding of the transformation student undergo.

The characteristics of the liminal space given by Meyer and Land, applied to our empirical data, served as starting points for an analysis on conceptual learning in computer science. This research has the potential to shed light on why some students get stuck at the threshold in the process of becoming computer scientists. Meyer and Land write: "liminality, we argue, can provide a useful metaphor in aiding our understanding of the conceptual transformations students undergo, or find difficulty and anxiety in undergoing, particularly in relation to notions of being 'stuck'." [13, p. 377]

3. RESEARCH QUESTIONS

This paper addresses two research questions.

1. Can the liminal space as discussed by Meyer and Land serve as a "useful metaphor in aiding our understanding of the conceptual transformations students undergo" [13, p. 377] in computer science?
2. What specific characteristics do we observe in computer science students when they are in the midst of learning a threshold concept, and do these satisfy the requirements of a liminal space?

By addressing these questions, we hope to gain better insights in the complicated process of conceptual learning in computer science. Furthermore we hope to shape the framework for our specific discipline.

We looked for evidence of the different features of a liminal space given in Meyer and Land's definition (above). It follows from the definition of a threshold concept that by learning it, the student is being transformed and acquiring new knowledge. Because threshold concepts are core concepts within a discipline, students learning them can also be said to be acquiring a new identity, that of an insider, someone who understands the central ideas of a field.

We focused, therefore, on the remaining aspects of the definition of a liminal space, looking for answers to the following questions in our interviews:

- Does the process of learning threshold concepts take time? Do the students appear to oscillate between the old and new states (i.e., not understanding and understanding)?
- What emotional reactions do students express?
- Does the process of learning threshold concepts involve mimicry?

We also formulated some questions that are not explicitly addressed by Meyer and Land, but notwithstanding seem to be important for a rich description of the learning of the threshold concepts studied.

- What kinds of partial understanding knowledge do students possess within the liminal space?

In the interviews we found that students made a clear distinction between different aspects of the concept they were learning. Most students discussed one or several aspects as troublesome to learn, but different students struggled with different aspects. Each of these aspects is a place where students might become stuck.

The second question we added was:

- Do students know that they have crossed a threshold, and if so, how?

Whether students can tell when they have crossed a threshold is relevant, since the liminal space seems to be accompanied by emotions of frustration or desire to pass through it. If a student thinks he or she has crossed a threshold in learning, even though he or she hasn't, what are the consequences for the motivation to learn?

4. RELATED WORK

This project fits squarely within the constructivist tradition. Constructivist theory holds that the learner actively builds knowledge. Different theories propose different models for the learner's knowledge: a hierarchy of anchoring ideas [11], schemas [11], and mental models. [7] In each case, however, learning involves adding to or modifying some cognitive structure. To continue the construction metaphor, threshold concepts are keystones, critical parts of the structure that hold the rest together, and the liminal space is the construction site.

No work has specifically addressed the liminal space in computer science. There is a substantial literature on concept learning in general, however. We will restrict ourselves to the work that is most closely related to the defining features of a liminal space.

Perkins and Martin found that students were hindered by what they called "fragile knowledge," that is, when the student "sort of knows, has some fragments, can make some moves, has a notion, without being able to marshal enough knowledge with sufficient precision to carry a problem through to a clean solution." [16, p. 214] Shymansky et al. found support for *oscillation* – "a punctuated, saw-toothed, conceptual growth process" – in a study of a group of middle-school teachers. [17, cited in [18]] In a later study of students, they found that while oscillations were not reflected in the mean ratings, 10 of the 22 individual students did show patterns of progress and regression. [18]

Mimicry is generally considered to be negative – the students are said to be "just mimicking" or "only mimicking" what they have seen. And it can be negative if the student does not progress beyond this point. Hughes and Peiris [6], for example, found a strong negative correlation between course performance and a "surface apathetic approach" to learning to program, in which the students memorize and reproduce what they have seen without any deeper understanding.

On the other hand, if students persist in seeking a deeper understanding while they mimic what they have seen, the practice may be helpful. In some non-Western educational traditions mimicry is considered to be an important step in learning. [8, 9] In a comparative study of Chinese and Australian students of accounting, Cooper found that "While surface approaches to learning can be associated with mechanical rote learning, memorization through repetition can be used to deepen and develop understanding and help achieve good academic performance." [2, p. 306]

Murphy and Tenenberg address the general question of *whether computer-science students know what they know.* [14] They asked students to predict how they would do on a data structures quiz taken in courses that required data structures as a prerequisite. They found that the students' estimates correlated moderately with their performance, and (interestingly) the accuracy of their estimates improved after the quiz. This was consistent with, or slightly better than, the estimating ability of students in other fields.

Mead et al. propose a method of *organizing a curriculum around what are essentially threshold concepts*, plus some additional "foundational concepts." [11, p. 187] They modify the definition of threshold concepts, requiring only that a concept be integrative and transformative. It seems likely, however, that the other defining features of threshold concepts follow from these two. If a concept integrates other ideas or causes you to see the field in a new way, it may well be troublesome to learn, and you are not likely to forget it. Thus, the set of concepts they focus on likely include all the threshold concepts.

They suggest creating a directed graph with the threshold and foundational concepts as nodes, showing the order in which concepts should be presented in a curriculum. Concept A should be taught before another concept B if it "carries part of the cognitive load in learning it." [11, p. 187] By presenting the concepts in the right order, we may be able to make it easier for our students to learn each of them.

Meyer and Land note that there is are inherent conflicts between the use of threshold concepts, particularly as articulated above by Mead et al. as steps in a logical passage through a curriculum, and the more fluid and unordered aspects of liminality on which we will focus here. [13, p. 379-380]

5. METHODOLOGY

The data used were gathered during a previous study of threshold concepts [1, 5, 10],using semi-structured interviews with 14 students at six institutions in Sweden, the United Kingdom, and the United States. For analysis, the student interviews were transcribed verbatim; where necessary, they were translated into English by the interviewer.

During the analysis of the data we identified two threshold concepts: object-orientation and pointers. [1] This paper continues the analysis by looking into how the idea of a liminal space relates to these threshold concepts. The authors once again read through all the interviews looking for quotes related to liminal space, the resulting selections were then discussed among the authors and related to the discussion of liminal space as described in Section 2.2. The result of this analysis is reported below.

In our interviews we specifically asked the students about concepts they found troublesome to learn. In the present study we have re-analyzed those interviews where students entered deeply into discussions on pointers or object-oriented programming.

6. RESULTS

The analysis was inspired by the goal to investigate the usefulness of the liminal space metaphor in computer science. We call our analysis a triangular conversation, that is an ongoing conversation and negotiation between the researchers, the data, and the liminal space as it is described by Meyer and Land. The questions we asked are inspired by the characteristics of the liminal space, but also by the data, the observed characteristics from the quotes. The answers we found are shaped by the research questions, the data, and our lengthy experiences as teachers in the subject domain.

6.1 Partial understanding

Since the liminal space for a concept is the time when the student is trying to attain a concept but has not yet succeeded, it should be characterized by partial attainment of a concept. When looking at the quotes relating to partial understanding, we see a number of common themes emerge. Students identify a number of different sorts of understanding: the abstract concept, concrete implementations using the concept, design using the concept, the rationale behind using the concept, the application of the concept to new problems, and mappings from one of these understandings to another. The observed understandings could be placed into these general categories:

- An abstract (or theoretical) understanding of a concept;
- A concrete understanding—the ability to write a computer program illustrating the concept—without having the abstract understanding;
- The ability to go from an understanding of the abstract concept to software design or concrete implementation;
- An understanding of the rationale for learning and using the concept; and
- An understanding of how to apply the concept to new problems—problems beyond those given as homework or lab exercises.

Abstract understanding

Some quotes showed that the abstract concept was not yet attained. This quote showed a confusion between the notions of *class* and *object*:

> *Subject 9:* I can still remember that I tried to do operations on the classes that I think I can really remember, but I think I was trying to let the lamp shine or don't shine by doing something with the lamp class instead of with the lamp object.

Another showed the difficulty of learning pointers was tied in with other unlearned concepts:

> *Subject 4:* You know, and I'm not sure what I didn't understand, because there was plenty of other things that we were doing at the same time, like recursion and inheritance, that also used pointers. Recursion was another huge stumbling block for me. And so taking a pointer and throwing it in with a recursive function - [laugh] - I felt like I was, you know when you stand in front of those mirrors in a dressing room, the ones that are in front of you and on the sides, and you see reflections and reflections and it never ends? That's what I felt like with pointers and recursion.

Some showed that some understanding had been successfully attained:

> *Subject 6:* But now I can ... I'm able to see how the classes are related, I guess. How they're related and which classes share information.

Some students were knowingly striving for a deep understanding:

> *Subject 5:* Why and how it [OOP] should be used. I have a background that is very much procedural imperative ... programmed Basic, Assembler and Pascal since the middle of the -80s ... so it was a rather high threshold to, not to learn how to use it, to get it to work, but to use it the right way ... I thought it was very elegant but it took probably several years before I saw the really elegant solutions ...

Concrete understanding without abstract

Some students were able to work with object-oriented concepts at a concrete level without a theoretical understanding:

> *Subject 9:* ... I'm pretty good at Java, but the interface concept is little strange. Abstract class and interface and stuff like that, ehh, is rather complicated. Ahh, specially interface [giggle]. And to explain that to someone, I don't think I can do it, but I can use the term and I can use interfaces.

Relating the abstract concept to implementation or design

Students commonly mentioned that they had a theoretical understanding, but were unable to translate that understanding to something less abstract. Specifically, many students discussed their inability to use their abstract understanding to produce a concrete implementation:

> *Subject 7:* There's just some aspects to it that just seem to remain kind of mysterious to me at the programming level. Not the concept level, not the theory level, not the technology level but at the kind of code nuts and bolts level ... It's not that I don't understand what I'm trying to accomplish it's just getting the syntax of the details right ... I'm a lot better in Java because I don't have to deal with the syntax of the details. I can only deal with the concepts.

> *Subject 8:* the abstract understanding is something you learn by education, by reading, you can learn that in class, but the understanding of actually applying it to programs you can't, you must, you must learn it by, by, by using it ...

Other mappings proved difficult, such as the application of the abstract understanding to design, which is less concrete than implementation:

> *Subject 8:* ...but the harder thing is actually to create what should be an object and what should not be an object and, what classes should I have for these things and, and that is the most ...

A similar observation is the following:

> *Subject 9:* ... So it wasn't like a big struggle to understand the difference between class and object for me actually. But it can be when you're designing a program ... To know where to stop doing the classes and start doing the objects. That ... that's actually something you can think about today, as well.

Rationale

Another aspect of understanding a concept is understanding its rationale–why you would want to know and use this concept. Students reported feeling the lack of this understanding as they were learning:

> *Subject 3:* My thoughts were that I didn't understand why we needed pointers when references worked perfectly well beforehand. I didn't understand the power of pointers and I guess I just didn't see the purpose of declaring variable `int*`[2].

> *Subject 5:* I found it difficult during the first ... the first course when I encountered it, I couldn't see the use of it, except that you could get some kind of encapsulation.

Application

Another way of understanding a concept is to understand how to apply it to new problems, not just in the related assignments.

> *Subject 2:* You understand how a theory works but how do you take that theory and how it works and apply it to a practical sense? I think that is one of the hardest leaps to make.

> *Subject 8:* ... it took a long time to understand how object oriented programming works, but then once I understood it more or less, the basic concept, I still couldn't use it, it wasn't usable because I didn't know what to apply to my problems.

6.2 Temporal Aspects and Oscillation

We examined how long students spend in the liminal space. Since we started our discussions by asking students about places where they were stuck, it is not surprising that all of our students emphasized the prolonged process required to learn threshold concepts.

Students and faculty alike will often talk about having an "aha" moment. While this might imply a sudden insight, this moment frequently is preceded by either a long time in the liminal space or a depth of understanding in a related area. Subject 7 implies a lengthy journey through the liminal space:

> *Subject 7:* It unwound or wound it printed out statements but I still didn't understand it very well. It really honestly wasn't until I got to your class that the light kind of came on and the idea of doing the checks up front and sort of assuming it's going to do what you tell it to do.

Subject 5 describes a deep understanding before the "aha" moment:

> *Subject 5:* A friend that showed me some kind of interpreter for some small little ... well, a model of a computer, where he [...] took the instruction object and told it 'run', [...] then I got some kind of small aha-experience, 'perhaps you can do it that way instead of doing it in some more tiresome way', the way I should have done it myself.

Many students mentioned a prolonged time in the liminal space. Across the board, the time to gain understanding was lengthy:

> *Subject 13:* I think there was definitely a point where I definitely got the understanding, whether I was still confident in doing it, that probably took a lot of time.

> *Subject 2:* ... when I finally did make the understanding, which actually took about two to three years.

> *Subject 6:* I think it is something that takes at least a couple of semesters. I mean, unless you've had prior experience, I just don't see.

> *Subject 5:* Object oriented programming was one thing for example that took a long time before ... it clicked. [...] It took ... perhaps two years before it was completely in-place ...

Students seemed to not only spend time learning the concepts, but they also demonstrated understanding that the prolonged process was necessary for learning. Subject 4 knowingly gives a lot of time to the learning process, while Subject 7 implies it was natural to not understand *yet*:

> *Subject 4:* So, I had a lot of time to spend, you know, brain resources to spend understanding, you know, stuff that's pointing and how you dereference it.

> *Subject 7:* So everybody - it was just sort of like they were talking a language I wasn't fluent in yet.

Students also implied oscillation in their learning. They describe the nature of going back and forth between knowing and not knowing, thinking that they know it, but realizing they're not there yet:

> *Subject 4:* It was clear to me, it just seemed like while I was in the thick of it I would forget. I spent a lot of time lost in the - it was that forest for the trees. I don't know. Lost in the jungle.

[2] `int*` is a reference to how a pointer to an integer is declared in C-like languages

Subject 6: ... with object-oriented [...] I think you understand the basic - you have the concept of it. But I ran into certain things with classes where I didn't have access to that particular class and I'm thinking, What's the problem here? [...] So I did understand but I have run into problems and it did kind of go back to objects and how they're relating.

Subject 9 nicely summarizes the oscillation between the knowing and the not knowing:

Subject 9: ...most of the time it's just iterates on the outside of the knowledge spiral. [...] I have to refresh the knowledge I learned recently more often, but some things I have to go back and refresh maybe the real basics of what it's all about. Not that I have forgotten it but to get a deeper understanding.

6.3 Emotional reactions

Meyer and Land refer to the liminal space as "problematic, troubling, and frequently involv[ing] the humbling of the participant." [13] They also warn that students may experience "difficulty and anxiety" in relation to learning threshold concepts. We examined our student quotes from this perspective, to see if we could find evidence of emotionally laden terms.

Students frequently mentioned that they found that learning threshold concepts was frustrating:

Subject 3: Felt? I think I felt frustrated. My thoughts were that I didn't understand why we needed pointers when references worked perfectly well beforehand.

Others referred to feelings of depression:

Subject 13: During ... well if I found it difficult then I would probably mope slightly for a while and then got down to it.

There was evidence of students feeling humbled:

Subject 2: Another thing that was very frustrating. I'm usually quick to understand things.

Subject 7: It just seems like it's been such a long and horrible road over pointers and that object oriented thing. That's just been my nemesis the whole way through and I don't remember anything else being that difficult.

Students themselves note that there was a certain mystique around becoming a programmer or understanding a concept:

Subject 7: The class idea was just really mysterious.

Subject 4: ... it seemed like something really hard. Like if you're extremely smart then you can program, you know. All computer geeks are really smart and they can program. That's my sort of opinion of it before I started. Something that was magical and hard.

The experience was definitely transformative if students eventually grasped a concept:

Subject 4: While I was stuck they [pointers] were a nightmare and I hated them. After I figured them out, they were very cool and useful. And I could see why you would want to have them.

Subject 6: And then when I do get it to work, it's almost like these people that run a 25-mile marathon just for, like, that high or whatever. I get that when I solve the problem. I get real souped, screaming in my room.

Confidence can be seen as an emotion with a fairly complex relationship with liminal spaces–being stuck can lower it, but having it can make it easier to get unstuck. Student 9 describes his or her feelings about the importance of having some prior knowledge in programming:

Subject 9: I got the aha experience again and that just was like if I know a little then I can, eh, jump in everywhere and catch up from there. [...] And that's really important to know, or to feel, that you can catch up. [...] it's not impossible.

The same student also emphasizes the importance of knowing "how to study" in terms of be able to use different online resources and IDEs:

Subject 9: That helped in Java doc and API on the Internet and the, aha experience again, that how to use it [...] that you could actually go there and see how you should do with any question you have and also seek information on other eh on Google for example on the Internet how to solve a specific problem, problem in programming. Eh, that have really helped me a lot, Ahh, the confidence that I can do it with, eh, help of Internet. [...] when I hear of a new concept it's just to see on the website to see what they mean and how they, how you should use it and you use it. Ah, and that's a really big threshold to come past.

6.4 Mimicry

During the interviews some of the students mentioned that in beginning to learn a subject they "imitated" someone or some existing code. Subject 9 discusses starting object-oriented programming:

Subject 9: And then ... when ... to learn something from an example, for example, it had to be exactly almost the same example as the thing you are trying to solve. You are trying to find exactly the information how to solve this problem in the textbook always search in the textbook.

and:

Subject 3: I think if a person can see the pattern, I think I'm no different from anyone else. If I can see the pattern, I can generally, I can take a technique and I can go home and figure it out if there's a pattern to it. I understand the pattern, why the pattern fits, and I can see how to figure out the exceptions to those patterns.

Others indicated that it was a big help in the beginning to have step-by-step instructions to follow:

> *Subject 7:* I think the idea - and one of the things in your teaching - one of the things about your teaching is that you tend to give a procedure and I think I don't believe in - you actually in some cases give a list of things. Step one, step two like on recursion. I don't think in other classes that we've been that procedure oriented. Maybe we've talked more about the idea and the concept and the whatever, but it really helped.

Even if a behavior like this might seem counter-productive, "the students are here to learn how to do things themselves, right", it is important to realize that for the interviewed students the "mimicry" seemed to be just a stepping stone in their further learning. Here is another example of a student who started by mimicking and then progressed:

> *Subject 9:* In the beginning I tried to look it up in the textbook and find the exact example how to solve this instead of, eh, while during the process I found that an IDE can help me when I, when I press the point it gets a list of everything that's possible to do with that object, and if you write the class name, you, you get some sort of error message, it probably meant that instantiation of this object. Ah, so it helps ...
>
> *Interviewer:* So, in this way ... are you using this IDE. Your understanding of these concepts changed?
>
> *Subject 9:* They improved, yes.

In some cases, however, students did not progress beyond mimicking:

> *Subject 7:* I have so much trouble with that overload asterisk and there's that – is it asterisk ampersand symbol or whatever. Never got that. Never had a clue. I just copied it. Yeah, it really gave me trouble. Just looking at would just sort of freeze me.

6.5 Crossing the threshold

Students in the interviews discuss object-oriented programming and pointers from the perspective of having passed the liminal space. This is expressed in different ways. Sometimes the descriptions of the experience of passing through the liminal space is emotional:

> *Subject 2:* It took a lot of just practicing and just repeating. It's to the point where when you see it you wouldn't be kind of intimidated. You would already say okay I know what I can do with this.

One student discusses the emotions that characterizes his or her conviction of having passed the liminal space, and the previous emotional conviction of not having passed:

> *Subject 6:* But I just remember at that moment like it just kind of made, I don't know, made sense, I guess. I don't know what about it made sense. [...] I mean, I did get it before. I saw what was going on. But I just didn't feel like I had the control, I guess, till I saw it.

Some students describe their conviction of having passed the liminal space as being able to visualize their understanding:

> *Subject 2:* But the basic idea of passing by reference or value; no, once I understood that I – every time it's mentioned I immediately know and understand – I can see a picture – a diagram in my head of what I'm supposed to do.
>
> *Subject 7:* I remember in the final I looked at a problem that you wrote and I saw recursion ... I remember it was a tree and I remember looking at it and as I said some people see black and white, some people see color. It was like I saw color. Oh, you can solve this with recursion. It's a tree. I can solve this recursively and here's this relationship. [...] That was kind of like, "Whoa." I actually saw it and that was pretty exciting.

Other students describe their conviction of knowing the concepts on the foundation of mastering the handicraft of programming:

> *Subject 13:* And after ... its like you said before it was one of those things like riding a bike, isn't it.

And another student says:

> *Subject 2:* And then after it's almost like it's a tool and you don't even think about using it. You say I need to do this. Okay, done. [...] And it's a seamless integration. It's just there it is. And you don't - it's almost like you don't even think. Like when you - right now I have to declare an integer. I don't think about how I do it or how to syntax. I just type it away. It's almost like a memory response.

The same student contrasts his or her experience to how it was before the passage of the liminal space:

> *Subject 2:* So I think it comes from a point of being completely lost and just randomly guessing and hoping your guessing is good. To a point where you're confident with using that and you may not want to use it as much as you would something else you're more confident with ...

Having passed the liminal space does not always mean that there is never a need for going back. The students distinguish however between not understanding and practicing for a better understanding, and between understanding but still needing to practice syntactical details of the programming language:

> *Subject 3:* After a certain length of time, yeah, sure, I have to review stuff.
>
> *Interviewer:* Well, you review it to program, but conceptually?
>
> *Subject 3:* No, I understand it. It's something I do get.

Similarly:

> *Subject 2:* I would have to look up the syntax and possibly get a very brief example just to remind myself that's how the pointer works. Okay, done. Then the memory jog hits me and I'm good.

An interesting question arises when studying students who claim they have passed through the liminal space. Are the students' views are in line with the educators' view of what is required for a "good" understanding of the concept? Are there students who believe they have passed the liminal space, when they, according to the course requirements, have not? And, on the other hand, are there students who believe they have not passed the liminal space, while educators would say they have? Students from our study illustrate this:

> *Subject 9:* So then, and still, I, I mean that I'm pretty good at Java, but the interface concept is little strange. Abstract class and interface and stuff like that, is rather complicated. 'specially interface. And to explain that to someone, I don't think I can do it, but I can use the term and I can use interfaces.

And later in the interview:

> *Subject 9:* I think I should know why information hiding is important but I can't think of it now ...

It can be questioned whether the student has passed the liminal space or not since the concepts the student fails to understand are central to the object-oriented paradigm.

Another student demonstrates his or her understanding of object-oriented programming, and yet says

> *Subject 5:* object oriented programming was one thing for example that took a long time before...it clicked. [...] It took...perhaps two years before it was completely in-place...and it's really nothing that I've really understood even today.

Reading the transcript as educators, we believe the student has a good understanding, and still he or she is not convinced of having passed the threshold.

rns out to be only partial. If the learning process is allowed to continue, the student might experience that he or she is back in the liminal space. The students' experience of passing the liminal space is thus not as reliable as it seems to be at the first glance. We want to point to the finding that there seem to exist students who think they have passed the liminal space, and thus probably do not strive for a better understanding, which is comparable with the discussion on misconceptions. We believe that our finding is an important aspect of the liminal space that might need to be considered in future work.

7. DISCUSSION

Students certainly describe the features that define liminal space according to Meyer and Land. Our analysis has raised a number of interesting observations and questions.

First, we saw different partial understandings of students during the liminal space. Students, at least in retrospect, show an appreciation that full understanding includes a number of aspects: abstract, concrete, rationale, application, and the connections among them. The need to attain all of these somewhat independent understandings explains why students get stuck at different places, and why the path through this space is not a simple linear progression. That we commonly observed the particular partial understanding of not being able to translate from an abstract understanding to concrete implementation or design may be specific to computing as a discipline, a question worth deeper investigation.

Second, when considering the question – "Does the process of learning threshold concepts take time?" – the answer seems to be a clear *Yes*. All of our subjects at some point discuss the lengthy process of learning. What we found interesting here was that whether acknowledging that learning occurs as a spiral action or as feeling lost in a jungle, all of our graduating students admit and accept that learning computing concepts takes time. This may be a major roadblock to first-year students who typically have not yet learned about the time-consuming nature of learning, particularly in a technically demanding field such as computing.

If so, then one thing educators can do is to support students during the experience that learning takes time. While many of us attempt to do this indirectly with our assignments and labs, is there something we can directly do about this? How can we instill the notion that the time-consuming nature of learning is normal?

This should also be taken into consideration by educators when they meet novice students. The insight that these students lack the experience that learning takes time might help educators to better understand and cope with the difficulties novice programmers demonstrate.

Third, we found that there was no lack of emotional reactions while learning threshold concepts. As our interviews show, students exhibit very strong feelings. This presence of strong emotion in students discussing the field of computing is rarely mentioned in the literature, but as CS educators, many of us have had the experience of students telling us that they "hate programming." Despite purists' belief that computing concepts should not be anthropomorphized [4], our students personalize threshold concepts, and say they *hate* or *fear* them. They also exhibit feelings of euphoria when they emerge on the other side of the threshold.

We suggest that instead of dismissing students' emotional reactions, as teachers and professionals we should recognize that they are normal and desirable. We need to consider how we can create a learning environment where the feelings that programming is hard, magical and frightening are handled, and students move through them rather than give up.

Fourth, many students state that at some stage during the learning process they mimic what others have done without exactly understanding what they are doing. For some teachers mimicry is an undesirable action; students are supposed to always understand what they are doing and to "just mimic" someone is a failure. We suggest that teachers look at mimicry in another way. Although some students do not progress past mimicry, it can be a step to gaining a full understanding of the subject. Meyer and Land acknowledge this when they write

> ...students might well adopt what appears to be a form of mimicry as a serious attempt to come to terms with conceptual difficulty, or to try on certain conceptual novelties for size as it were. We would not want to belittle or dismiss such responses as they may well prove to be success-

> ful routes through to understanding for certain learners. [13, p. 383]

Our original interviews did not pursue the idea of mimicry in depth, but the mixed results here suggests that further study of the role of mimicry in learning programming (and computing in general) is warranted.

Fifth, the question "Do students know that they have crossed a threshold, and if so, how?" has no clear answer. We have identified different ways students express their belief that they have passed the liminal space. The descriptions are often vivid and illustrate the students' experiences of a transformation. Yet, while students express that they understand a subject, the evidence suggests that they might be wrong. What are the consequences if students think they have passed through the liminal space, when they have not? And, on the other hand, how does it affect students if they believe themselves not to have reached desired understanding, when the educator judges that they already have passed the liminal space?

8. CONCLUSIONS

In this study, we examined Meyer and Land's notion of liminal spaces in the context of learning concepts in computer science. Addressing our research questions, we found

1. Liminal spaces provide a useful metaphor for the concept learning process, at least for transformative concepts. The absence of a single path through, the fact that these changes can take time, the emotional reactions of the students, and the students using mimicry as a coping mechanism: these characteristics seem to capture much of the learning experience.

2. In addition to observing the "standard" features of liminal spaces, we have identified some that may be specific to computing. The kinds of partial understandings observed—specifically those relating to levels of abstraction—are closely tied to what computer scientists need to do.

The particular emphasis on the difficulty in going from the abstract to the concrete is quite interesting, and seems counter-intuitive given the way we teach computing. The emphasis in computer science education is on teaching students to abstract away from the details, but the problems observed here are in moving in the other direction.

The most important practical observation from this work may be that different students take different routes through the liminal space, with the possibility of getting stuck at multiple places. This suggests that there is no fixed order of topics that best serves all students, rather instruction should be flexible enough to accommodate individual students. Knowing what aspects of a concept are necessary to gain full understanding, particularly those concerning different abstraction levels and the mappings from more to less abstract, could help here.

This work suggests a number of questions that deserve further investigation. Would we get similar results if we interviewed students while they were still in a liminal space, rather than after they have attained understanding? Would we see differences if we interviewed novices rather than graduating seniors? Would other transformational concepts in computer science—those less tied to programming, for example—show similar partial understandings? How do students use mimicry when trying to learn, and when might it be effective?

Stay tuned.

ACKNOWLEDGMENTS

The authors would like to thank Mark Ratcliffe and Jonas Boustedt, who participated in the design, data collection, and initial analysis of the threshold concept interviews. Thanks also the Department of Information Technology at Uppsala University for providing us with workspace and facilities in Uppsala, and to Sally Fincher, Josh Tenenberg, and the National Science Foundation (through grant DUE-0243242) who provided workspace at the SIGCSE 2006 conference in Houston. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Uppsala University.

9. REFERENCES

[1] J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, K. Sanders, and C. Zander. Threshold concepts in computer science: do they exist and are they useful? In *SIGCSE-2007*, pages 504–508, Covington, KY, March 2007.

[2] B. J. Cooper. The enigma of the Chinese learner. *Accounting Education*, 13(3):289–310, 2004.

[3] P. Davies. Threshold concepts: how can we recognise them? 2003. Paper presented at EARLI conference, Padova. http://www.staffs.ac.uk/schools/business/iepr/docs/etcworkingpaper(1).doc (accessed 25 August 2006).

[4] E. Dijkstra. On the cruelty of really teaching computing science. *Commun. ACM*, 32(12):1398–1404, 1989.

[5] A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, K. Sanders, and C. Zander. Putting threshold concepts into context in computer science education. In *ITiCSE-06*, pages 103–107, Bologna, Italy, June 2006.

[6] J. Hughes and D. R. Peiris. ASSISTing CS1 students to learn: learning approaches and object-oriented programming. In *ITiCSE-06*, pages 275–279, Bologna, Italy, June 2006.

[7] P. N. Johnson-Laird. *Mental models: towards a cognitive science of language, inference, and consciousness*. Harvard University Press, 1983.

[8] D. Kember. Misconceptions about the learning approaches, motivation and study practices of asian students. *Higher Education*, 40:99–121, 2000.

[9] F. Marton, D. Watkins, and C. Tang. Discontinuities and continuities in the experience of learning: An interview study of high-school students in Hong Kong. *Learning and Instruction*, 7(1):21–48, 1997.

[10] R. McCartney, A. Eckerdal, J. E. Moström, K. Sanders, and C. Zander. Successful students' strategies for getting unstuck. In *ITiCSE-07*, Dundee, Scotland, UK, June 2007. (To appear).

[11] J. Mead, S. Gray, J. Hamer, R. James, J. Sorva, C. St. Clair, and L. Thomas. A cognitive approach to identifying measurable milestones for programming skill acquisition. In *ITiCSE-WGR '06: Working group*

reports on ITiCSE on Innovation and technology in computer science education, pages 182–194, New York, NY, USA, 2006. ACM Press.

[12] J. Meyer and R. Land. Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines. ETL Project Occasional Report 4, 2003. http://www.ed.ac.uk/etl/docs/ETLreport4.pdf.

[13] J. H. Meyer and R. Land. Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher Education*, 49:373–388, 2005.

[14] L. Murphy and J. Tenenberg. Do computer science students know what they know?: a calibration study of data structure knowledge. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 148–152, New York, NY, USA, 2005. ACM Press.

[15] D. Perkins. The many faces of constructivism. *Educational Leadership*, 57(3):6–11, 1999.

[16] D. N. Perkins and F. Martin. Fragile knowledge and neglected strategies in novice programmers. In *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers*, pages 213–229, Norwood, NJ, USA, 1986. Ablex Publishing Corp.

[17] J. A. Shymansky, G. Woodworth, O. Norman, J. Dunkhase, C. Matthews, and C.-T. Liu. A study of changes in middle school teachers' understanding of selected ideas in science as a function of an in-service program focusing on student preconceptions. *Journal of Research in Science Teaching*, 30:737–755, 1993.

[18] J. A. Shymansky, G. Woodworth, O. Norman, J. Dunkhase, C. Matthews, and C.-T. Liu. Examining the construction process: a study of changes in level 10 students' understanding of classical mechanics. *Journal of Research in Science Teaching*, 34(6):571–593, 1997.

[19] V. Turner. *From Ritual to Theatre: the human seriousness of play*. Performing Arts Publications, New York, 1982.