Threshold Concepts and Threshold Skills in Computing

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

Jonas Boustedt
Faculty of Engineering and Sustainable Development
University of Gävle
Gävle, Sweden
jbt@hig.se

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

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

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

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

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

ABSTRACT

Threshold concepts can be used to both organize disciplinary knowledge and explain why students have difficulties at certain points in the curriculum. Threshold concepts transform a student's view of the discipline; before being learned, they can block a student's progress.

In this paper, we propose that in computing, skills, in addition to concepts, can sometimes be thresholds. Some students report finding skills more difficult than concepts. We discuss some computing skills that may be thresholds and compare threshold skills and threshold concepts.

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, threshold skills

1. INTRODUCTION

While learning computing, students acquire both theoretical concepts and practical skills. Some of these theoretical concepts may act as "threshold concepts": ideas within a discipline whose learning fundamentally changes the learner's viewpoint of that discipline. The idea of threshold concepts has been useful as a way to both organize disciplinary knowledge and explain why students have difficulties at certain points in a curriculum. [7]

Most investigations of threshold concepts in computing to date ([15] for example) have identified potential concepts that are effectively aspects of learning to program. In our own investigations [35], we identified pointers as a potential threshold concept in computing: we observed that students exhibited the characteristics of being in a liminal state as they acquired this concept [18], a state in which they struggled to apply their knowledge by writing code. Programming involves conceptual knowledge, but it is also a skill that improves with experience – something you do, not just a concept you know.

When we interviewed students about threshold concepts, they often emphasized problems with learning how to apply the concepts through programming rather than the concept itself. One interpretation of this: *The acquisition of particular skills, and not the concepts related to these skills, can act as thresholds.* This interpretation, plus descriptions of aspects of programming in the literature, led us to re-examine threshold concepts interview data with a focus on skill acquisition, to see whether Threshold Skills exist, and (if so) to see how they compare to Threshold Concepts.

The structure of the paper is as follows. In Section 2 we review three things: skills, knowledge, and threshold concepts. In Section 3 and Section 4 we more formally state the research questions that this paper addresses and discuss our methodology. Our findings are discussed in Section 5:

what the analysis says about threshold skills in computing. Section 6 presents some quantitative results about retention in a course where the students gain experience applying concepts that they previously learned. Section 7 looks at the relationship between threshold skills and threshold concepts and possible implications of these results. Finally, in Section 8 we provide conclusions drawn from this work and suggest further work that might follow.

2. BACKGROUND AND RELATED WORK

Our examination of threshold skills comes from our belief that writing computer programs is a skill, not simply a collection of concepts: an effective programmer knows the concepts and also knows *how* to write programs. This section discusses skills, knowledge – the relationship between theoretical and practical knowledge as well as the distinctions between declarative and procedural knowledge – and threshold concepts.

2.1 What is a skill?

Terms such as "skill" have been widely used in the discussion of what students should be learning, but without a clear consensus on their definition. [9] For purposes of this paper, we start from the *Oxford English Dictionary* definition:

> Capability of accomplishing something with precision and certainty; practical knowledge in combination with ability; cleverness, expertness. Also, an ability to perform a function, acquired or learnt with practice. [27]

In particular, the second part of this, the *ability to perform a function, acquired or learnt with practice*, is consistent with our experiences with things like programming.

Examples of skills are easy to find. Ryle [30] suggests making omelettes, conducting battles, designing dresses, persuading juries, and running experiments. While some conceptual knowledge is needed for all of these in varying degrees, intuitively, they primarily involve non-conceptual knowledge – knowledge that cannot easily be written down in the form of propositions and communicated – and all are best learned by practice.

Using the above definition, we can identify many skills in our interview data. Some are involved in programming such as using pointers, recursion, and data structures; others are non-programming skills such as object-oriented design and writing reduction proofs.

There exists a considerable body of research on students' learning programming skills like reading and writing code [21, 17], debugging [14], and software design [11]. These studies all report that students have severe problems in developing their skills.

2.2 Epistemology - what is knowledge?

When discussing skills, we cannot avoid discussing epistemological aspects: what is knowledge? The concept of knowledge has been discussed using different and partly overlapping terminology in different contexts. We give a summary of aspects of these discussions, including our own use.

2.2.1 *Theoretical and practical knowledge*

Research in professional education has considered the different kinds of knowledge using the terms *theory* and *practice*, with the understanding that a professional needs to master both area-specific practices and their related theories. However, the view on the relationship between theoretical knowledge and practical skills in educational contexts varies. In Western culture there is agreement that theory and practice are opposite parts of a dualistic opposition; knowledge is regarded as either theoretical or practical [16, 24]. In discussing the quality of student learning, Entwistle [13, p. 3] writes: "In the student learning literature, there has been an emphasis on conceptual understanding to represent high quality learning, but this had to be broadened to cover additional skills and ways of thinking, both academic and professional." There is a longstanding debate on the relationship between theory and practice in higher education. Dewey argued in the beginning of the 20th century, that theory should be the foundation of professional education [32], but at the same time he sought "intelligent practice" [8, p. 125]. He rejected theory without deep understanding as well as practice learned as procedures removed from their meaning. Shulman comments on Dewey's research saying: "he argued that only theoretical learning *situated in practice* would be rich and meaningful" [32, p. 524, italics in original].

The complex relation between students' learning of concepts and their learning of practical procedures in the lab is seen in science education, see e.g., [31, 1]. In computer science education it is generally accepted that students need to learn both theory and practice to become skillful professionals. It is further widely acknowledged that not only the theory but also the practice can cause great problems in students' learning. For example, du Boulay [4] discussed domains that programming students must learn to master, including the syntax and semantics of a programming language and different programming skills.

2.2.2 *Declarative and procedural knowledge*

In mathematics education research the terminology "conceptual" (or declarative) versus "procedural" knowledge is frequently used. McCormick [20, p. 149] explains that conceptual and procedural knowledge relate to "a familiar debate in education, namely that of the contrast of content and process . . . In mathematics education the argument has been about 'skills versus understanding'." Procedural knowledge is often discussed as less advanced and preceding conceptual knowledge.

In the language-instruction domain, there exists a comparable dichotomy between declarative and procedural knowledge:

> Language teachers and language learners are often frustrated by the disconnect between knowing the rules of grammar and being able to apply those rules automatically . . . [M]any native speakers can use their language clearly and correctly without being able to state the rules of its grammar. Likewise . . . students may be able to state a grammar rule, but consistently fail to apply the rule when speaking or writing. [25]:

2.2.3 *Knowing what and knowing how*

We are making a distinction between knowing *how* and knowing *what* similar to that in the philosophical literature. Knowing what includes the kind of conceptual understanding that has been discussed as threshold concepts. We focus on knowing how, as described by Ryle:

> *(a)* When a person knows how to do things of a certain sort (e.g., make good jokes, conduct battles or behave at funerals), his knowledge is actualised or exercised in what he does. ...His intelligence is exhibited by deeds, not by internal or external dicta.
>
> *(b)* When a person knows how to do things of a certain sort (e.g., cook omelettes, design dresses or persuade juries), his performance is in some way governed by principles, rules, canons, standards or criteria. ...But his observance of rules, principles, etc., must, if it is there at all, be realised in his performance of his tasks. [30, p. 8]

Ryle's description of the difference between knowing how and knowing what is a philosopher's; a similar idea is discussed by Norman in the context of engineering design:

> Knowledge *of* – what psychologists call declarative knowledge – includes the knowledge of facts and rules. ...Declarative knowledge is easy to write down and to teach.
>
> Knowledge *how* – what psychologists call procedural knowledge – is the knowledge that enables a person to perform music, to stop a car smoothly with a flat tire on an icy road, to return a serve in tennis, ...Procedural knowledge is difficult or impossible to write down and difficult to teach. It is best taught by demonstration and best learned through practice. [26, pp. 57-58]

Both Norman and Ryle provide support for the notion that concepts and skills are different. Moreover, Ryle provides some examples of skills that are non-physical – designing dresses, persuding juries, and playing chess. [30]

2.3 Threshold Concepts

The theory of Threshold Concepts was developed by Meyer and Land [22, 23] as a way to understand the learning of concepts that change the way a learner views a discipline. A subset of the core concepts in a discipline, threshold concepts are described as having the following characteristics:

- *transformative*: they significantly 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.
- potentially *troublesome* (as in [28]) for students: they are conceptually difficult, alien, and/or counter-intuitive.
- probably *irreversible*: they are difficult for the student to unlearn.
- often *boundary markers*: they indicate the limits of a conceptual area or the discipline itself. [22]

Generally speaking, Threshold Concepts are "conceptual gateways" (or barriers): things the student needs to understand to go further in his or her discipline. Examples given in [22] include *limits* and *complex numbers* in mathematics, and *opportunity cost* in economics.

In previous work, we identified three computing concepts that fit the description of Threshold Concepts: abstraction (including the ability to move flexibly from one level of abstraction to another), object-orientation, and pointers. [10, 5] These papers did not discuss skills, although many of the student interviews we empirically build on involve an element of skill as well as conceptual knowledge.

Another characteristic of Threshold Concepts is that the learner can find him- or herself in a "liminal state" while learning [23], characterized by an incomplete change in perspective, or an oscillation between the old and new perspectives. In [12] we found that computing students' experiences are consistent with those in [23]: the students may take significant time being transformed, they may react emotionally, and they sometimes use mimicry as a coping mechanism. Moreover, the analysis identified different kinds of partial understandings that students possess within the liminal space:

1. An abstract/theoretical understanding of a concept;
2. A concrete understanding – the ability to write a computer program illustrating the concept – without having the abstract understanding;
3. The ability to go from an understanding of the abstract concept to software design or concrete implementation;
4. An understanding of the rationale for learning and using the concept; and
5. An understanding of how to apply the concept to new problems.

If we consider software design and implementation to be a skill, then the second of these is an example of having the skill without the conceptual understanding, and the third and fifth illustrate connections between skills and conceptual understanding.

Previous research has indicated that learning certain skills might exhibit characteristics akin to learning threshold concepts. From computer science, Sien and Chong [33] propose object-oriented modeling to be a threshold concept. From mathematics education, Worsley, Bulmer and O'Brien [34] propose that the technique of substitution and solving ordinary differential equations are candidates for threshold concepts. These problems seem close to our view of skills as thresholds. However, none of the articles pursue the question how these skills relate to threshold concepts.

3. RESEARCH QUESTIONS

This paper addresses the following research questions:

- Are there skills whose acquisition acts as a threshold for computing students?
- If yes, how do these "threshold skills" compare to threshold concepts? Specifically, what characteristics do they share (and fail to share) with threshold concepts?

4. METHODOLOGY

For this project, we re-analyzed a set of semi-structured interviews concerning threshold concepts, looking for references to skills. In addition, we examined quantitative data regarding student performance in a course where a significant part of the learning involved new skills, while the related concepts were already familiar to the students.

4.1 The interviews

The interview data consist of sixteen semi-structured interviews on the topic of threshold concepts. Participants included graduating 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.

Fourteen of these interviews used a script that began by addressing the troublesome criterion, asking students for concepts they found difficult at first (places where they were initially "stuck"). From these, we selected one concept to pursue in depth and addressed the transformative, integrative, and irreversible criteria in that context. We do not examine the boundary marker criterion in detail, since our interviews were only about core concepts and did not focus on the disciplinary boundary of computer science. These interviews formed the basis for a previous study of threshold concepts [10, 5, 19, 18], and the full interview script is found in [5].

To further explore the concept of transformation, we conducted two additional interviews with a variant of the original script in which we explored the transformative criterion first. The questions in the script were modified as needed to fit the new order. In these additional interviews, the participants spent more time discussing transformation and less time talking about how troublesome a concept was, how it felt to be stuck, etc. With regard to skills, however, there was no notable difference.

4.2 Analysis of the interviews

This is an exploratory study. Until now, thresholds have been discussed in terms of concepts. We report here on a preliminary investigation of how, in computing, skills might be, or be part of, these thresholds. As a result, we chose to do a primarily qualitative analysis.

We began by reading and re-reading the interview data, looking for quotes related to skills. Once we were satisfied that skills played a significant role in most of the interviews, we conducted a deductive thematic analysis. [6] Each of the characteristics of threshold concepts – transformative, integrative, troublesome, and irreversible – was examined by at least two researchers. The researchers assigned to a particular characteristic re-read all the interviews and coded extracts related to that characteristic, considering them in relation to skills. During the analysis, the need to practice emerged as an aspect of skills, so two researchers examined that as well.

4.3 The quantitative data

In Section 6, we discuss quantitative data that provide some further support for the hypothesis that skills can be thresholds. These data quantify student performance in a course required for all computing students at the institution where one of the authors teaches. They were gathered from 2005-2011.

5. QUALITATIVE FINDINGS

In this section, we present the ways in which the threshold concept characteristics are manifested in the context of skills as well as the relationship between skills and practice.

5.1 Transformative

Mastery of a threshold concept transforms a student's perception of his or her discipline. Mastering a threshold skill transforms what students can *do* – and their vision of what they can do. They can solve problems they couldn't solve before or solve them in a better way.

In discussing transformations, the students often emphasize their newly acquired skills. For example, Student 15 reports that after taking data structures, he could do more, and he *knew* he could do more.

> [I began to] really think about it [programming] on a deeper level of how I could do things ... faster ... seeing how I can make all those data structures that are in use, and then if I wanted I could make my own or combine them. I did that over the summer for an internship. And I couldn't have done that before. I wouldn't have ever thought of that.

True mastery of a skill means it becomes almost automatic. Before Student 2 mastered the skill of writing C++ code with pointers, for example, he'd "randomly try something different until it works, which sometimes I thought I had the answer and sometimes I didn't." He wasn't "really sure *how to use it*" [emphasis added]. Afterwards, "then 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."

Sometimes there is a complex relationship between a threshold skill and a concept. For example, Student 6 had heard in lecture that object-oriented programs are made up of a collection of cooperating objects and had practiced writing object-oriented code for some time, when he finally got it. "It was just the relation between the two. This class was just going to do this job and that's it. And this class just needed that information. So it accessed the other class." This change in understanding was accompanied by a significant change in his perception of what he could do: "[I]t was, like, sitting down, I could add my own class. I could come up with stuff and just go nuts with it on my own." In this case, the threshold seemed to include both a concept and a skill, tightly connected together.

Student 1 went a step further and reflected on the process of transformation. He noted that he had learned several design patterns, and each time, the process was the same. He struggled for a while, then understood why the pattern was useful and how it worked, and eventually became comfortable using it. Find the quote in Section 8.

5.2 Integrative

Learning a threshold concept is integrative, in that it exposes connections between concepts in the discipline that were previously unknown to the learner. There seems to be an analogous notion with threshold skills, but related to their application to tasks or problems: once a threshold skill is attained by applying it to some task, the learner sees other places where the skill could be applied. These other places can be tasks that the user could already perform, but using different techniques. Student 1, for example, was able to find recursive solutions for a lot of tasks that he had done before using iteration, and in procedural as well as functional languages. Student 2 discusses how the application of a skill to different problems can lead to this sort of integration:

> [T]he only way to make that is really just through doing it multiple times or seeing examples where it's done. And saying oh, I can use it this way. Oh, I never would have thought of that. ... And then when it's off that base knowledge you can start making different connections.

Gaining a skill in one context may lead to improved understanding in others. Student 4 had problems dereferencing pointers in C++, but figured out how to do it when doing problems in assembly:

> I can't explain why assembly made it more clear than C++. The only thing I can think of is that in assembly you rely on those references heavily, because you don't have a ... whole lot of registers ... you can't declare too many variables.

5.3 Troublesome

Threshold concepts can be difficult for students as they are associated with troublesome knowledge (as in [28]): knowledge that is inert, ritual, conceptually difficult, or foreign. Although the researchers were trying to elicit troublesome concepts in these interviews, the students frequently discussed skills. They reported having difficulties not when they listened to a lecture or read the text, but when they had to *do* something. Students report this problem in a variety of different contexts. Student 14 talked about the difficulty of learning to program:

> I remember from the first year how when I was going into the lectures I could see the way that you were doing things and I could mentally work it out for myself, but when it came around to doing it ... it was if sometimes I was hitting a brick wall.

Similarly, Student 2 reported that he understood how recursion "theoretically worked" but still, "writing an application" took time, and Student 4 talks about the difficulty of writing code using pointers:

> So, before pointers I thought programming wasn't that hard. And then I got into pointers and the doing part; I was just lost.

The students may have understood the concepts and been unable to apply them, or perhaps they didn't fully understand them until they could apply them. In either event, however, they definitely perceived applying the concept to be the troublesome aspect of learning.

5.4 Semi-irreversible

One of the features used to describe a threshold concept that it is probably irreversible; once learned it is difficult to "unlearn" or even to "think back" to how a concept was viewed before understanding. Irreversibility seems somewhat weaker with skills – skills can degrade over time through lack of use, but they do not completely go away. The student stays transformed, and knows where the skill applies, but it may be necessary to review or practice to regain or maintain skills.

Student 7, for example, discussing pointer manipulation and the need to review:

> I haven't written a C++ program in a while so if I was going to go back and code that again I'd have to dig out my old code and kind of review and look at it. I'm not sure that I can create it from scratch on the fly.

Student 4 agrees and describes how if she is "doing it every day then if you ask this question, [I] can explain it a lot better."

We saw some examples of skills that seemed to be irreversible once attained. Student 10 remembered having difficulties when learning object-oriented programming, but is unable to explain why from his current perspective:

> It just made so much sense to me right now that it's hard to think back at how I didn't understand it.

5.5 Associated with practice

As suggested by the definition in Section 2, skills are closely related to practice: part of the definition of a skill is that it is attained or learnt through practice, where practice is "repeated exercise in or performance of an activity or skill so as to acquire or maintain proficiency in it."[27] In our data, students describe using practice in both ways: to initially acquire a skill, and to maintain proficiency.

While learning a skill, students talk about the need for repetition or drill, to try many examples. For example, Student 3 gives advice and discusses his experience learning to use pointers:

> ... draw lots of pictures. For example, write a lot of programs that have lots and lots of pointers. Write lots of linked lists and manipulate them as much as you possibly can. Dereference them and find out, do a lot of `cout` statements and find out exactly where you are in memory. ...
> I made sure that the address was exactly what it was supposed to be pointing to because then I would get the address, I would go into the debugger and get the address of the object...
> Yeah, I saw it update and all that stuff. Yeah. I saw when it got to a null value and I changed my code. It went out of bounds. I made all the mistakes.

Student 10 thinks that practicing your skills is the essential thing that creates engineers:

> I think really the engineers need to learn that to be successful in engineering; to be successful in life you need to somehow sometimes just buckle down and just really work at it for hours – slave away at it.

The use of practice to maintain skills is closely related to whether threshold skills are irreversible.

6. QUANTITATIVE SUPPORT

Here we provide support for the notion of threshold skills seen in formal education, where teachers observe that it is the nuts and bolts of programming that stymies the students. Section 5 discusses skills as thresholds with evidence obtained from student interviews. Here we see direct support of programming skills as a threshold. We report on a third-year C++ course where students are expected to previously have learned fundamental programming concepts. We present empirical evidence that the students have not yet acquired the associated programming skills.

At one author's university, students start out in typical CS1 and CS2 courses at the home university or at a regional college or university. The majority learn Java; fewer than 5% have seen C/C++. To be admitted to the program, students must have had success in CS1 and CS2: receive a grade of B- or better (greater than 2.7 on 4.0 scale).

The first required course in the third year, is somewhat a repeat of CS2 conceptually, but in C++. Concepts taught in CS2, elementary abstract data structures such as linked list, stacks, and queues are used as foundations; algorithm complexity is formalized and recursion is revisited. A focus of the course is on pointers, memory management, and object-oriented design and programming in C++. There are discrete math topics not necessarily covered in their CS2 course: induction; introduction to propositional and predicate logic; recurrence equations. While topics other than programming are covered, the focus is on programming with the mantra by faculty that a student must be a competent programmer to be able to succeed to the next course.

Students in this course report that they do not find the programming concepts new. Yet, it has been repeatedly observed that they struggle with using pointers and memory management. The statistics (available numbers from different terms offered 2005 through 2011), seen in Table 1, support the observed difficulty with the skill.

Course Offering	Drops	Inadequate Grade	Number of Students	Not Successful
1	5	2	30	23.3%
2	5	1	22	27.3%
3	4	3	34	20.6%
4	2	9	26	42.3%
5	5	14	51	37.3%
6	3	6	39	23.1%
7	11	8	73	26.0%
8	6	2	36	22.2%
Totals	41	45	311	27.7%

Table 1: Students not successful using C++

Only students who dropped the course after performing poorly on the midterm exam are counted in Table 1. These students would likely not have succeeded with a satisfactory grade. Students completing the course with unsatisfactory grades to continue to the next course (less than 2.0 on a 4.0 scale) typically are inadequate at programming.

This course highlights the threshold skills of using pointers and managing memory. The students were successful in CS1 and CS2, but many, nearly 30%, still have difficulties crossing this skill threshold. Students are confused by pointers, references, pointers to pointers, and references of pointers. While the concept that memory holds an address is straightforward, students struggle with implementation, with the application of the concept.

7. DISCUSSION

While we do not have a definitive answer, our data suggest that the answer to our first research question, "Are there skills whose acquisition acts as a threshold for computing students?" is yes. Different aspects of threshold skills are mentioned in relation to various skills and, as shown in Section 5, the skill of *using pointers* shows up in relation to each of the threshold skill criteria.

Transformative: When learning pointers, a student went from making "random" changes to code in order to make it work, to being able to use pointers routinely, to solve problems he couldn't solve before.

Integrative: Understanding pointers in assembly language helped a student to understand how to use them in C++.

Troublesome: a students describes how when he started using pointers, he went from thinking "programming wasn't that hard" to being "completely lost."

Semi-irreversible: Students say they could use pointers even though it had been some time since they had done so – but it would require review.

Associated with practice: A student's advice on learning pointers includes many practice activities: drawing lots of pictures, dereferencing many pointers, verifying addresses in the debugger, and so forth.

From these data, it appears that using pointers is an example of a skill that exhibits both threshold concept and skill attributes. Students emphasize that the skill of using pointers, not the concept, is the threshold.

In answer to our second question, threshold skills are both like and unlike threshold concepts.

Threshold concepts are transformative, integrative, and irreversible. A good example is user-centered design: the notion that software is designed for other people to use. Introductory students don't generally think in this way (and judging by the software on the market, not all computing professionals do either). The idea, once grasped, transforms the way students view their profession; it affects many aspects of software design, development, and testing; and because it is such a fundamental shift in perspective, it is unlikely to be reversed.

Threshold skills can be transformative, integrative, and (semi-)irreversible – but in different ways from a threshold concept. A skill is **transformative** if it changes what students can do – and their vision of what they can do. It is **integrative** if it can be used in different contexts. Sometimes it broadens the list of problems students can solve; sometimes it gives them a new and perhaps better way to solve problems they could already handle. It is **semi-irreversible** if it can be regained with practice, without having to start again from scratch. It is rarely learned in a single aha moment; generally it must be acquired with practice.

Threshold skills and concepts can both be **troublesome** in much the same ways. Some skills, such as recursion, may be perceived as *foreign*; without the perspective of how functions are invoked, a function calling itself may seem like nonsense. A skill can be *inert* if a student has the ability to use it in a narrow context but not in others: for example a student who uses stacks effectively in a data structures course but fails to use them when they apply to problems in a later course. A skill cannot be *conceptually difficult*, but it can be complex and demanding to learn and maintain.

It may be that skills are more likely to be thresholds in computing than many other disciplines. When students learn the concepts related to their native language in elementary school, such as nouns, sentences, and paragraphs, they already have a skill – speaking the language – to which they can tie these concepts. In many university-level subjects, the students build upon skills developed in primary and secondary education. When our students learn computing, they are generally learning skills and concepts at the same time. This may be especially challenging since knowledge of skills is partly tacit. Polanyi writes: "*There are things that we know but cannot tell.* This is strikingly true for our knowledge of *skills*." [29, p. 601, italics in original]

An implication of this is that we may need to identify the skills that are hard for students to attain, and to allocate more time in our instruction for practice. Students may listen to the lectures, read the book, believe that they understand – and maybe really understand – and still be unable to apply a concept. Students having difficulty with C++ even though they have demonstrated competence with Java supports this notion. As a discipline, perhaps we need to develop drills comparable to the scales played by musicians or athletes' practice.

Another implication is that acquiring skills is particularly important for retention. Whether or not students understand the related concept, what they focus on is the skill. There is external feedback: the program doesn't compile, the recursion causes a stack overflow, or the program simply doesn't do what it's supposed to do. As a result, lack of a skill not only blocks their progress, but frustrates and discourages them in an immediate way that failure to understand a concept might not.

Finally, we have observed that the problems related to gaining skills does not only apply to beginning students: more advanced students report similar problems e.g., with pointers or design patterns. Skills, therefore should also be considered in more advanced courses. One may ask whether the traditional assignment and assessment forms in Computer Science (problem solving and projects) cover the individual student's needs for gaining the appropriate skills.

8. CONCLUSIONS AND FUTURE WORK

In this paper we looked at skills as thresholds, suggesting that their learning exhibits analogous characteristics to those seen with threshold concepts. Here is a quote from Student 1 that displays several of the characteristics of threshold skills discussed earlier, with our annotations in square brackets and italics:

> There's like a big gap between the theoretical good software design, and the how do you implement it well *[The difference between concept and skill]*. And I found design patterns are a nice way for me to bridge that gap *[Integrates theory and practice]*. But it takes, you know, a month or so of just working with one to really understand when it's appropriate, how to use it to its full advantage *[Need for practice to attain skill]*. And then if I don't use it regularly, I forget how to structure it *[Need for practice to maintain skill]*.

This student appreciates that skills are important in computing as well as concepts.

Our research into threshold concepts has shown that what students describe as problematic about learning a concept is not always understanding the concept in its abstract sense. They often think they understand the general meaning – the principle. Instead, the perceived difficulty is to handle the details of specific applications. It may be interesting to consider this in relation to the Biggs [2] and Bloom [3] taxonomies. To recognize and re-tell something is less advanced than using it. To draw useful conclusions from the experiences of using something is more sophisticated than just using it.

What does it mean to know a concept? To become truly familiar with a concept in programming, it is not enough to know the conceptual idea, its principles and rules: one must also be able to use the concept unhindered in its applications – to make things with the concept. Only then the concept is understood in its deeper meaning. Hence, we found it interesting to look the combination of concepts and the requisite skill.

We still do not know if threshold skills always relate to one or more threshold concepts. In our previous research we suggested pointers as a threshold concept candidate. Following from the present findings we now further suggest that pointers also exhibit the characteristics of what we describe as a threshold skill. We believe there is a need for further research in this area. We hope that the threshold concept theory will provide a useful framework to study how these skills are learned and maintained, and how they complement conceptual computing knowledge in our students.

9. REFERENCES

[1] C. von Aufschnaiter and S. von Aufschnaiter. University students' activities, thinking and learning during laboratory work. *European Journal of Physics*, 28(3):51–60, 2007.

[2] J. B. Biggs and K. F. Collis. *Evaluating the quality of learning : the SOLO taxonomy (structure of the observed learning outcome)*. Academic Press, New York, 1982.

[3] B. S. Bloom, editor. *Taxonomy of Educational Objectives: the Classification of Educational Goals. Handbook 1: Cognitive Domain*. Longman, New York, 1956.

[4] B. du Boulay. Some difficulties of learning to program. In E. Soloway and J. Spohrer, editors, *Studying the Novice programmer*, pages 283–299. Lawrence Erlbaum Associaties Inc., 1988.

[5] 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? *SIGCSE Bull.*, 39(1):504–508, 2007.

[6] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3:77–101, 2006.

[7] 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).

[8] J. Dewey. Science as subject-matter and as method. *Science*, 31(787):121–127, 1910.

[9] C. Dörge. Competencies and skills: Filling old skins with new wine. In N. Reynolds and M. Tursanyi-Szabo, editors, *Proceedings of the key competencies in the knowledge society: IFIP TC 3 International Conference, KCKS 2010*, pages 78–89, 2010.

[10] 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, 2006.

[11] A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, and C. Zander. Categorizing student software designs: Methods, results, and implications. *Computer Science Education*, 16(3):197–209, September 2006.

[12] A. Eckerdal, R. McCartney, J. E. Moström, K. Sanders, L. Thomas, and C. Zander. 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, NY, USA, 2007. ACM.

[13] N. Entwistle. Concepts and conceptual frameworks underpinning the ETL project. Occasional Report 3 of the Enhancing Teaching-Learning Environments in Undergraduate Courses Project, School of Education, University of Edinburgh, March 2003, 2003.

[14] S. Fitzgerald, G. Lewandowski, R. McCauley, L. Murphy, B. Simon, L. Thomas, and C. Zander. Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2):93–116, 2008.

[15] M. T. Flanagan and J. Smith. From playing to understanding: the transformative potential of discourse versus syntax in learning to program. In R. Land, J. H. F. Meyer, and J. Smith, editors, *Threshold Concepts Within the Disciplines*, chapter 7, pages 91–104. Sense Publishers, Rotterdam, 2008.

[16] J. P. A. M. Kessels and F. A. J. Korthagen. The relationship between theory and practice: Back to the classics. *Educational Researcher*, 25(3):17–22, 1996.

[17] R. Lister, T. Clear, Simon, D. Bouvier, P. Carter, A. Eckerdal, J. Jackova', M. Lopez, R. McCartney, P. Robbins, O. Seppälä, and E. Thompson. Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bulletin*, 41(4), 2009.

[18] R. McCartney, J. Boustedt, A. Eckerdal, J. E. Moström, K. Sanders, L. Thomas, and C. Zander. Liminal spaces and learning computing. *European Journal of Engineering Education*, 34(4):383–391, 2009.

[19] R. McCartney, A. Eckerdal, J. E. Moström, K. Sanders, and C. Zander. Successful students' strategies for getting unstuck. *SIGCSE Bull.*, 39(3):156–160, September 2007.

[20] R. McCormick. Conceptual and procedural knowledge. *International Journal of Technology and Design Education*, 7(1-2):141–159, 1997.

[21] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *SIGCSE Bulletin*, 33(4):125–180, 2001.

[22] 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, Universities of Edinburgh, Coventry, and Durham, 2003. http://www.ed.ac.uk/etl/docs/ETLreport4.pdf.

[23] 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.

[24] B. Molander. *Kunskap i handling [Knowledge in Action; in Swedish].* DAIDALOS, 1996.

[25] National Capital Language Resource Center (NCLRC). The essentials of language teaching: Strategies for learning grammar. `http://www.nclrc.org/essentials/grammar/stratgram.htm`. Accessed March 13, 2012.

[26] D. A. Norman. *The Design of Everyday Things.* Doubleday, New York, 1990.

[27] *Oxford English Dictionary Online.* Oxford University Press, March 2012. http://0-www.oed.com.helin.uri.edu/ viewdictionaryentry/ Entry/ 180865 (accessed April 07, 2012).

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

[29] M. Polanyi. Tacit knowing: Its bearing on some problems of philosophy. *Reviews of Modern Physics*, 34(4):601–616, 1962.

[30] G. Ryle. Knowing how and knowing that: The presidential address. *Proceedings of the Aristotelian Society, New Series*, 46:1–16, 1945.

[31] M. Séré. Towards renewed research questions from the outcomes of the european project *Labwork in Science Education. Science Education*, 86(5):624–644, 2002.

[32] L. S. Shulman. Theory, practice, and the education of professionals. *The Elementary School Journal*, 98(5):511–526, May 1998.

[33] V. Sien and D. W. K. Chong. Threshold concepts in object-oriented modelling. In *7th Educators' Symposium@MODELS 2011 - Software Modeling in Education - Pre-Proceedings*, pages 55–64, Carl von Ossietzky universität Oldenburg, 2011.

[34] S. Worsley, M. Bulmer, and M. O'Brien. Threshold concepts and troublesome knowledge in a second-level mathematics course. In *Symposium Proceedings: Visualisation and ConceptDevelopment, UniServe Science*, pages 139–144, The University of Sydney, 2008. UniServe Science.

[35] C. Zander, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, and K. Sanders. Threshold concepts in computer science: a multi-national investigation. In R. Land, J. H. F. Meyer, and J. Smith, editors, *Threshold Concepts Within the Disciplines*, chapter 8, pages 105–118. Sense Publishers, Rotterdam, 2008.