Putting Threshold Concepts into Context in Computer Science Education

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 06269 USA
robert@cse.uconn.edu

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

Mark Ratcliffe
Department of Computer Science
University of Wales
Aberystwyth, Wales
mbr@aber.ac.uk

Kate Sanders
Department of Math and Computer Science
Rhode Island College
Providence, RI USA
KSanders@ric.edu

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

ABSTRACT

This paper describes Threshold Concepts, a theory of learning that distinguishes core concepts whose characteristics can make them troublesome in learning. With an eye to applying this theory in computer science, we consider this notion in the context of related topics in computer science education.

Categories and Subject Descriptors

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

General Terms

Theory

Keywords

Threshold Concepts, Education research, Constructivism

1. INTRODUCTION

As educators, we are aware of topics that are difficult for our students to learn, yet necessary for their development as computer scientists. Meyer and Land [25, 26] have proposed using what they call "Threshold Concepts" as a way of characterizing particular concepts that might be used to organize and focus the educational process. These are concepts whose achievement is necessary to making progress in the discipline, that transform the way a student looks at a discipline, but are also places in the curriculum where students get stuck, unable to make progress until they become unstuck.

In this paper, we look at the defining characteristics of Threshold Concepts in general, then place them in the context of related work in the computing discipline.

1.1 Characteristics of Threshold Concepts

Threshold Concepts are a subset of the core concepts in a discipline. Core concepts are building blocks that must be understood; Threshold Concepts, in addition, are [25]:

1. *transformative*: they change the way a student looks at things in the discipline.
2. *integrative*: they tie together concepts in ways that were previously unknown to the student.
3. *irreversible*: they are difficult for the student to unlearn.
4. potentially *troublesome* for students: they are conceptually difficult, alien, and/or counter-intuitive.
5. often *boundary markers*: they indicate the limits of a conceptual area or the discipline itself. Students who have mastered these Threshold Concepts have, at least in part, crossed over from being outsiders to belonging to the field they are studying.[1]

These criteria are all closely related: if something truly transforms the way you look at your discipline you are unlikely to forget it; if something integrates concepts in previously unknown ways it transforms the way you see those

[1]In listing the criteria for Threshold Concepts, some have used the term "bounded" for the concepts themselves.It seems clear, however, that what is meant is that the discipline, or part of it, is bounded *by* the Threshold Concept, and the concepts are boundary *markers.* For example, Davies states, "[A] threshold concept is bounded. That is, it helps to define the boundaries of a subject area." [8, p. 5].

topics; and the process of understanding something that is alien or counter-intuitive may well require a mental transformation.

These concepts, then, are thresholds that our students must cross, but where many of them get stuck. They are both conceptually difficult and lead to a broader (and necessary) understanding of the discipline. If we can identify these concepts, they can provide focus points within the curriculum: places where we might place extra effort to help students avoid getting stuck, and thus continue to make progress.

Particular Threshold Concepts have been suggested in a number of disciplines: *limit* in Mathematics, *opportunity cost* in Economics, *irony* in Literary criticism, *depreciation* in Accounting, e.g. Identifying such concepts in different disciplines may indicate whether this is a generally useful approach or one that applies only to fields with certain characteristics. We are interested, naturally, in computer science.

While the notion of Threshold Concepts is new to computer science[23], it relates to a number of other CS education research areas. In this paper, we consider some related areas of research, and show how these relate to (and differ from) Threshold Concepts. In particular, we discuss the following:

- constructivism, particularly in CS education;
- mental models;
- student misconceptions;
- breadth-first approaches to introductory computer science, an approach built around important computing concepts;
- Fundamental Ideas, an alternative organizational framework based on concepts that develop and persist through the curriculum; and
- two specific concepts, abstraction and object-orientation; specifically why they are good candidates for computer science Threshold Concepts.

2. CONSTRUCTIVISM

Constructivists interpret student learning as the development of personalised knowledge frameworks that are continually refined. According to this theory, to learn, a student must actively construct knowledge, rather than simply absorbing it from textbooks and lectures [9]. Students develop their own self-constructed rules, or "alternative frameworks"[1]. These alternative frameworks "naturally occur as part of the transfer and linking process"[7]. They represent the prior knowledge essential to the construction of new knowledge[32]. When learning, the student modifies or expands his or her framework in order to incorporate new knowledge.

Constructivism is a theory of learning; Threshold Concepts are points where students have difficulty learning. In constructivist terms, Threshold Concepts are distinctive parts of a student's knowledge framework, parts that make a computer scientist's knowledge framework different from that of someone who has not studied computer science. They connect other parts of the framework together, and they are parts of the framework that are particularly troublesome, or difficult to build. The difference between students' and instructors' knowledge frameworks, like Threshold Concepts, help to explain the "knowledge barrier" that exists between students and instructors. Threshold Concepts are core learning outcomes about which there is some general agreement in the discipline, so they may be more objective than knowledge frameworks; on the other hand, it may well be that some students experience a given concept as a threshold, and others do not.

Threshold Concepts, as developed by Meyer and Land, are closely tied to the constructivist tradition. Indeed, their use of "troublesome" follows from Perkins [28] discussing challenges that constructivists must face.

3. MENTAL MODELS

A theory of how our brains work when we reason about the world, well known in cognitive psychology, is that we form mental models [19]. Briefly, a mental model is one person's internal model of a system's properties and behavior. The use of a mental model makes it possible to predict the system's response to various actions and thus makes it possible for an individual to select the best possible action [20].

The consequence of this is that a "faulty" mental model can lead to "faulty" actions. A classic example: to increase the temperature in a room as fast as possible a common action is to turn the room thermostat up as much as possible, based on the expectation that a thermostat works the same way as a water tap. However, since a thermostat works like a switch, the temperature increases at the same rate as long as the setting is above the temperature of the room. Similarly, a faulty model of how various programming constructs work will cause problems when developing software.

Ben-Ari [1] argues that the lack of mental models plays an important part in why students find it difficult to learn how to program. His argument is that, having no previous models to build on, programmers are forced to construct their own mental models from scratch. Wiedenbeck [34] investigated how novice programmers' mental models of their programs depend on whether a procedural or an object-oriented language was used. Similarly, Yehezkel et al [35] describe the importance of forming a mental model of a system in order to understand it. Wiedenbeck et al [33] claim that the ability to form mental models is a predictor for course outcome.

In these terms, Threshold Concepts are points at which students may have trouble forming their mental models. Mastering the relevant threshold concepts may make it possible to form a mental model, or change one mental model into a new one. For example, a possible Threshold Concept could be the difference between "pass-by-value" and "pass-by-reference", understanding this difference will have a profound impact on how a person would understand the semantics of a programming language and model the data flow in a program.

While there are similarities between threshold concepts and mental models – both can for example be transformative – there are also major differences. Threshold Concepts are troublesome to learn while some mental models can be learned without much effort, for example modeling the computer screen as a desktop. Threshold Concepts are accepted concepts within a discipline, while mental models are subjective and individual.

4. STUDENT MISCONCEPTIONS

Research into misconceptions, like Threshold Concepts, focuses on the ways in which students *fail* to learn. Over the past almost 30 years, this work has had considerable impact [32]. According to constructivist theory, misconceptions naturally occur as students modify and extend their knowledge frameworks to learn new topics. For example, an individual's previous understanding can lead to misconceptions when familiar terms are used in unfamiliar contexts. Bonar and Soloway [3] use the *while* statement to demonstrate the problem of linguistic transfer. In common language the *while* can imply continual testing of the condition (*e.g.*, "hold your breath while underwater"). In programming loops the time of the test is limited: it occurs once only on each iteration. Students who interpret the test as continual have a misconception. The overloading of language, mathematical symbols and previous programming experience, taking information from one context and using it in another have all been demonstrated to cause misconceptions [7].

Misconceptions can be integrative, like Threshold Concepts (although the integration is not necessarily correct), but can also result from failing to integrate knowledge. As Eylon and Lynn [14] have observed, students deal with apparent contradictions by keeping their knowledge isolated. This might explain why students often fail to transfer knowledge from one course to another.

Misconceptions, like Threshold Concepts, can be hard to forget. As students familiarize themselves with new topics, their partial knowledge leads them to develop their own rules [15]. Unfortunately because the knowledge is incomplete these self-constructed rules may result in misconceptions, which once established are difficult to change. These premature generalizations are then used to filter and distort new information often compounding the misconception [18]. In order to realign such robust ideas, significant effort is required, undertaking radical reordering of the concept [24].

Unlike Threshold Concepts, misconceptions are certainly not core concepts that we want our students to learn. Moreover, we can speculate that they are not troublesome to learn. Especially if the correct (threshold) concept is difficult or counter-intuitive, the misconception may be much easier to learn.

5. BREADTH-FIRST INTRODUCTION

Threshold Concepts are a subset of the core concepts in the discipline. Thus, a list of the core concepts would be a good starting point in identifying the Threshold Concepts.

One source for such a list is the breadth-first approach to teaching computer science, since such courses are often based on important ideas or topics in computer science The concepts covered vary from course to course, but can include decision trees, number representation, patterns in programming, divide-and-conquer, recursion, the Church-Turing thesis, the von Neumann architecture, time complexity, intractability, types and values, classes and objects, design, encapsulation, inheritance, polymorphism, program correctness, iteration, recursion, conceptual and formal models, levels of abstraction, reuse, and tradeoffs.[2, 5, 10, 30].

The only criteria for these concepts is that they be, in terms of the definition of Threshold Concepts, core concepts in computer science. While most computer scientists would agree on the general importance of these concepts, they are probably not all Threshold Concepts. A concept such as "divide and conquer," for example, integrates many areas of computer science, but it does not appear to be troublesome for students to understand. Similarly, the idea of reuse is an easy one to grasp (even though it may be harder to implement consistently). The concepts on the breadth-first courses' lists may not be transformative, and some of them are probably all too easy to forget. Which, if any, of them qualify as Threshold Concepts is a question for empirical investigation.

6. FUNDAMENTAL IDEAS

Schwill[31] has proposed organizing the computing curriculum around another set of core concepts, the Fundamental Ideas, a set of ideas that are central to the discipline. This follows from the work of Bruner [6], who proposed that science teaching should be organized around the structure of science, as expressed by its fundamental ideas. These are ideas with broad applicability, and that can be taught at multiple levels within the curriculum, from early to advanced, at increasingly sophisticated levels.

Drawing on work applying Fundamental Ideas to mathematics, Schwill proposes four criteria for these ideas in computer science (paraphrasing):

Horizontal criterion The idea is applicable or observable in multiple ways and multiple areas of CS; it organizes and integrates multiple phenomena.

Vertical criterion The idea can be taught at every intellectual level, at different levels of sophistication.

Criterion of time The idea is clearly observable in the history of computer science, and will be relevant for a long time.

Criterion of sense The idea also has meaning in everyday life, and can be described in ordinary language.

These can be used to organize and relate subjects in computer science. Fundamental Ideas are taught throughout the curriculum, and when new concepts are presented, they are related to the appropriate Fundamental Ideas that the students know, thus providing context. Moreover, relating new concepts to these ideas should further develop the ideas, so the learning process can be seen as gradually gaining a greater understanding of these Fundamental Ideas.

There is some obvious overlap between Fundamental Ideas and Threshold Concepts. Both are integrative, and both include topics that should be understood by any competent computing professional. But the Fundamental Ideas are likely *not* transformative, in that they are gradually developed from common-sense understanding of everyday phenomena. Threshold Concepts, on the other hand, may not be teachable at every level. Finally, Fundamental Ideas are clearly not boundary markers, given the *Criterion of sense*, as these ideas have everyday out-of-discipline meanings.

Overall, these approaches seem to be orthogonal. Threshold Concepts are based on transformative events, while Fundamental Ideas are based on long-term development. It seems likely that any given Threshold Concept could be described in terms of the related Fundamental Ideas, and that there are Threshold Concepts that appear at points in the development of a given Fundamental Idea. Threshold Concepts identify the discontinuities in a student's development, while Fundamental Ideas identify different ongoing threads in this process which may or not have such discontinuities.

7. PARTICULAR CONCEPTS

For illustration, we consider two particular candidate Threshold Concepts, abstraction and object-orientation, in relation to the criteria for Threshold Concepts.

7.1 Abstraction

The ability to abstract, and more than that, to move flexibly from one level of abstraction to another, is a key skill in computer science. As noted in Section 5, abstraction is a core concept, and and it has been widely studied. If one searches the papers available through the ACM Digital Library (which includes over 200 computing journals) using the keyword "abstraction", 63% of all articles are found.

Or-Bach and Lavy [27] construct a cognitive task analysis taxonomy regarding abstraction and inheritance. They find that abstraction is key with relation to object-oriented programming and determine that it is a higher order cognitive skill difficult for students to conceptualize.

Similar findings about students' use of methods and attributes with regard to abstraction were found also by Detienne [11] who claims that one of the main difficulties experienced by novices is the articulation between declarative and procedural aspects of the solution. While an object can be thought of as an abstract data type, in object-oriented design it seems appropriate to consider the abstraction inherent in object orientation as behavior abstraction. This understanding is seen in advanced OO designers, but not in novices.

Box and Whitelaw [4] argue that abstraction helps explain the relative difficulty of learning object-oriented technology. In this technology, the learning of new concepts include several steps, and abstraction is both the last and most difficult step. Significant parts of this abstraction are the decisions as to which entities are to be grouped together and which attributes are to be ignored or parameterized.

Hadjerrouit [17] writes about students' learning of the object-oriented programming language Java:

> ...to understand Java concepts properly, problem solving should begin at the conceptual level, not at the code level where programming becomes the main issue. Furthermore, substantial attention should be devoted to the meta-level process required to develop solutions.

Hadjerrouit expresses a constructivist perspective, and a clear emphasis on the importance of gaining an understanding of the abstract concepts as well as the concrete.

7.2 Object-orientation

Much has been reported on experiences with teaching the object-oriented paradigm. It is widely acknowledged that Object-oriented programming is difficult to teach [21].

Here we consider Object-orientation at its most basic: including *objects, classes,* and *encapsulation,* but ignoring such things as inheritance and polymorphism. Even at this simplified level, it is a core concept in computer science, the first one learned by many introductory students, and necessary for students to understand. Holland, Griffiths and Woodman [18] claim that misconceptions of object concepts can be hard to shift later. Such misconceptions can act as barriers through which later all teaching on the subject may be inadvertently filtered and distorted.

The literature suggests that this satisfies the requirements for a Threshold Concept. We consider these requirements in turn.

There is much evidence in the literature that students find basic object-orientation *troublesome* to learn. Eckerdal and Thuné [13] interviewed first year students who had just finished their first programming course on their understanding of the concepts object and class. Many students stated that they found the concepts troublesome to learn despite great effort to understand them. Ragonis and Ben-Ari [29] studied high-school students in a first programming course in Java. They found that one of the major difficulties is to understand the creation of object by constructor. Fleury [16] has investigated Java students comprehension of encapsulation and reuse of code. One result she reports is that many students consider reducing the number of lines of code and reducing the number of classes to be more important than encapsulation. Many students are annoyed by the "jumping around" necessary when reading programs with multiple classes.

There is also evidence that object-orientation is *transformative.* Luker [22] argues that learning the object-oriented paradigm, "requires nothing less than complete change of the world view". Eckerdal [12] reports that some students, who had used other programming paradigms before, could use the concepts object and class in a way that made programming more efficient.

Luker [22] furthermore discusses how encapsulation ties together the concepts object and class. Eckerdal [12] found that students had problems separating the concepts object and class. These suggest that basic object-orientation *integrates* these concepts, and gives some justification for our considering them together.

8. CONCLUSIONS

In this paper, we discussed the idea of Threshold Concepts as a possible way to organize and focus learning in computer science. Moreover, we tried to put it in context with other areas of computer science education. Summarizing,

> **Constructivism** seems to be assumed by Threshold Concepts: learning these concepts are particularly interesting places in the process.
>
> **Mental Models:** Threshold Concepts are places where fundamentally different mental models are developed, often with difficulty.
>
> **Misconceptions** The transformational nature (and difficulty) of attaining Threshold Concepts make them obvious places where misconceptions can be formed; observed misconceptions suggest places to look for Threshold Concepts.
>
> **Breadth-first introductions** involve teaching a broad range of "important" core concepts; as Threshold Concepts are also a subset of the core ideas there may be overlap.
>
> **Fundamental ideas** seem to be an orthogonal organization principle, emphasising the long-term develop-

ment of central ideas rather than just the transformational points.

Abstraction and Object-orientation are two possible concepts that may be Threshold Concepts; certainly evidence exists that they have the appropriate characteristics.

Threshold Concepts may prove to be useful in CS Education once we identify them. Previous work in Computer Science Education suggests places that we might reasonably start to look.

9. REFERENCES

[1] M. Ben-Ari. Constructivism in computer science education. *J. Computers in Mathematics and Science Teaching*, 20(1):45–73, 2001.

[2] A. Biermann. *Great Ideas in Computer Science: a gentle introduction*. MIT Press, 1990.

[3] J. Bonar and E. Soloway. Preprogramming knowledge: A major source of misconceptions in novice programmers. In E. Soloway and J. Spohrer, editors, *Studying the Novice Programmer*. Lawrence Erlbaum Associates, 1989.

[4] R. Box and M. Whitelaw. Experiences when migrating from structured analysis to object-oriented modelling. In *Proceedings of the Australasian conference on Computing education*, pages 12–18. ACM, 2000.

[5] J. G. Brookshear. *Computer Science: an overview*. Addison Wesley, sixth edition, 2000.

[6] J. Bruner. *The process of education*. Harvard University Press, Cambridge, MA, 1960.

[7] M. Clancy. Misconceptions and attitudes that interfere with learning to program. In S. Fincher and M. Petre, editors, *Computer Science Education Research*. Taylor and Francis Group, London, 2004.

[8] P. Davies. Threshold concepts: how can we recognise them. 2003. Paper presented at EARLI conference, Padova. Downloaded from http://www.staffs.ac.uk/-schools/business/iepr/docs/etcworkingpaper(1).doc.

[9] R. Davis, C. Maher, and N. Noddings. Constructivist views of the teaching and learning of mathematics. *J. Res. Math. Teaching, Monograph No.4*, 1990.

[10] P. Denning. Great principles in computing curricula. *SIGCSE Bull.*, 36:336–341, 2004.

[11] F. Detienne. Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams. *Interacting with Computers*, 9:47–72, 1997.

[12] A. Eckerdal. On the understanding of object and class. Technical Report 2004-058, Dept. of Information Technology, Uppsala Univ., 2004.

[13] A. Eckerdal and M. Thuné. Novice java programmers' conceptions of "object" and "class", and variation theory. In *Proc. ITiCSE '05*, pages 89–93, 2005.

[14] B. Eylon and M. Linn. Learning and instruction: An examination of four research perspectives in science education. *Rev. Educational Research*, 58(4), 1988.

[15] A. E. Fleury. Programming in java: student-constructed rules. *SIGCSE Bull.*, 32(1):197–201, 2000.

[16] A. E. Fleury. Encapsulation and reuse as viewed by Java students. *SIGCSE Bull.*, 33(1):189–193, 2001.

[17] S. Hadjerrouit. A constructivist framework for integrating the java paradigm into the undergraduate curriculum. *SIGCSE Bull.*, 30(3):105–107, 1998.

[18] S. Holland, R. Griffiths, and M. Woodman. Avoiding object misconceptions. *SIGCSE Bull.*, 29(1):131–134, 1997.

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

[20] D. E. Kieras and S. Bovair. The role of a mental model in learning to operate a device. *Cognitive Science*, 8:255–273, 1984.

[21] M. Kölling. The problem of teaching object-oriented programming, part I: Languages. *J. Object-oriented Programming*, 11(8):8–15, 1999.

[22] P. A. Luker. There's more to OOP than syntax. *SIGCSE Bull.*, 26(1):56–60, 1994.

[23] R. McCartney and K. Sanders. What are the "threshold concepts" in computer science? In *Proceedings of the Koli Calling 2005 Conference on Computer Science Education*, page 185, 2005.

[24] M. McCracken, W. Newstetter, and J. Chastine. Misconceptions of designing: a descriptive study. *SIGCSE Bull.*, 31(3):48–51, 1999.

[25] 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.

[26] J. 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.

[27] R. Or-Bach and I. Lavy. Cognitive activities of abstraction in object orientation: an empirical study. *SIGCSE Bull.*, 36(2):82–86, 2004.

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

[29] N. Ragonis and M. Ben-Ari. Teaching constructors: A difficult multiple choice. In *16th European Conference on Object-Oriented Programming, Workshop 3*, 2002.

[30] G. M. Schneider and J. L. Gersting. *An Invitation to Computer Science*. Brooks Cole, second edition, 1998.

[31] A. Schwill. Fundamental ideas of computer science. *Bull. European Assoc. for Theoretical Computer Science*, 53:274–295, 1994.

[32] J. Smith, A. diSessa, and J. Roschelle. Misconceptions reconceived: A constructivist analysis of knowledge in transition. *J. Learning Sciences*, 3(2), 1993.

[33] S. Wiedenbeck, D. LaBelle, and V. N. R. Kain. Factors affecting course outcomes in introductory programming. In *16th Annual Workshop of the Psychology of Programming Interest Group*, 2004.

[34] S. Wiedenbeck and V. Ramalingam. Novice comprehension of small programs written in the procedural and object-oriented styles. *Int. J. Human-Computer Studies*, 51:71–87, 1999.

[35] C. Yehezkel, M. Ben-Ari, and T. Dreyfus. Computer architecture and mental models. *SIGCSE Bull.*, 37(1):101–105, 2005.