Successful students’ strategies for getting unstuck

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

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

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

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

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

ABSTRACT
Students often “get stuck” when trying to learn new computing concepts and skills. In this paper, we present and categorize strategies that successful students found helpful in getting unstuck. We found that the students reported using a broad range of strategies, and that these strategies fall into a number of recognizably different categories.

Categories and Subject Descriptors  
K.3.2 [Computers and Education]: Computers and Information Science Education—Computer Science Education

General Terms  
Measurement, Experimentation

Keywords  
Learning strategies, stuck places, Threshold Concepts

1. INTRODUCTION
Learning does not always occur at a constant rate of increasing knowledge and skills. While learning new concepts and skills, students sometimes encounter epistemological obstacles [11]—that is, they get stuck and are unable to make progress toward learning and understanding.

In this paper we look at strategies that successful computing students reported using to become unstuck and make progress in learning these concepts and skills. These students are successful in two ways: one, they successfully learned particular computing concepts after being stuck, and two, they have been successful in their educational programs: the students we interviewed were within a year of graduation, all due to finish their degree programs by the end of 2006.

The goal of this investigation is twofold. First, we would like to identify strategies that students use successfully in their computing studies. Second, we would like to categorize these strategies in ways that make them useful for future students and instructors. We have approached this investigation from the student perspective, focusing on what students report about their learning.

Section 2 describes the techniques we have used to gather information about getting unstuck from the student’s perspective, the student strategies we identified, and a hierarchy into which those strategies can be organized. In Section 3 we provide examples from interviews that illustrate the range and depth of the students’ use of learning strategies. In Section 4, we compare our results to those reported in the general and computing education literature. In Section 5, we give some overall impressions of the way that students used these strategies, and how we might use this information in making instruction more effective. Finally, in Section 6, we present our conclusions and discuss the future directions of this research effort.

2. DATA GATHERING AND ANALYSIS
The data used here were gathered using semi-structured interviews, as part of a larger investigation into Threshold Concepts in computing [2, 4]. Fourteen students (total) were interviewed 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.

Some of the interview questions dealt explicitly with the idea of being stuck and becoming unstuck—these were used to identify possible threshold concepts to pursue in depth. The parts of the script dealing with these topics are given in Figure 1 (for a more complete script, see [2]). The students provided a surprisingly rich list of strategies they used to get unstuck, along with advice for other students in similar situations.
1. Could you tell me about something where you were stuck at first but then became clearer? (Subject answers <X>.)
   The rest of this session will now focus on <X>.
2. Can I start by asking you to tell me your understanding of <X>?
4. Tell me your thoughts, your reactions, before, during and after the process of dealing with <X>.
5. Can you tell me what helped you understand <X>?
7. Based on your experience, what advice would you give to help other students who might be struggling with <X>?
13. To finish the interview, can you tell me whether there are any other things where you were stuck at first but then became clearer?

Figure 1: Interview script excerpt: parts concerning stuck places, getting unstuck, and strategies.

Impressed by the students’ responses, we examined this portion of our data from a new angle. After extracting the quotes relevant to this topic, we proceeded inductively, identifying and naming 35 distinct strategies, grouping those into 12 more abstract categories, and finally grouping those 12 into four super-categories, all of which in turn are examples of Get unstuck/learn. We worked individually, discussed with the group until we reached agreement, and referred back to the original quotes as needed.

The 35 basic strategies, the 12 categories, and the 4 super-categories are shown in Table 1; the top levels of the hierarchy (from Get unstuck/learn on down) are shown in tree form in Figure 2.

3. WHAT THE STUDENTS SAID
Space does not permit inclusion of all the interesting quotes, or even quotes illustrating all 35 of the basic strategies. In this section, we give some of the most interesting quotations, illustrating each of the four super-categories.

3.1 Inputs/interaction
Many students talked about getting help from elsewhere. Not surprisingly, subjects read, looked information up on the Internet, and used tools. Subject12 says

... instead of basically doing it I would sit there and read trying to figure out how to do it ...

Subject9 demonstrates a common tendency to rely on tools:

... helped in Java doc and API on the Internet.

Subjects frequently learned from other people. Subject11 discusses getting the information needed to figure out problems from a variety of sources:

Like either be it peer or, you know, another faculty member that, you know, understands the problem.

It was suggested by Subject7 for instructors to give step-by-step instructions that students can follow until they are comfortable with the material:

So you could as a teaching tool, you could say, now we’re going to do this and these are the steps to putting these things together, just like you did with recursion. Step one, write out what you are going to do. Step two, write out what the defining check, whatever - those things. In this program we’ll do this, this, this and this. Just trust me. It’s going to work.

3.2 Concrete/do stuff
Often a single quote showed multiple strategies. Practicing and learning from examples was a common combination with getting help from others. Subject3 discusses the instructor, learning from examples, and practice:

Table 1: Identified strategies and their abstractions

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Abstract strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discuss</td>
<td></td>
</tr>
<tr>
<td>Learn from peers</td>
<td>Learn from other people</td>
</tr>
<tr>
<td>Listen to professor</td>
<td></td>
</tr>
<tr>
<td>Get help (from a person)</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>Learn from tools or written materials</td>
</tr>
<tr>
<td>Use a tool</td>
<td>Get and follow step-by-step instructions</td>
</tr>
<tr>
<td>Get and follow step-by-step instructions</td>
<td></td>
</tr>
<tr>
<td>Remember things</td>
<td></td>
</tr>
<tr>
<td>Be persistent/don’t stop</td>
<td></td>
</tr>
<tr>
<td>Avoid the problem / work around</td>
<td></td>
</tr>
<tr>
<td>Walk away and come back later</td>
<td></td>
</tr>
<tr>
<td>Reflect/sit and think</td>
<td></td>
</tr>
<tr>
<td>Write programs</td>
<td></td>
</tr>
<tr>
<td>Learn by trial and error</td>
<td>Gain experience</td>
</tr>
<tr>
<td>Learn from your mistakes</td>
<td></td>
</tr>
<tr>
<td>Practice/drill</td>
<td></td>
</tr>
<tr>
<td>Visualize/see a diagram</td>
<td>Visualize</td>
</tr>
<tr>
<td>Draw diagrams/pictures</td>
<td></td>
</tr>
<tr>
<td>Connect diagrams with code</td>
<td></td>
</tr>
<tr>
<td>Use examples (in general)</td>
<td></td>
</tr>
<tr>
<td>Use varied examples</td>
<td>Learn from examples</td>
</tr>
<tr>
<td>Use sequence of increasingly complex examples</td>
<td>Trace</td>
</tr>
<tr>
<td>Trace</td>
<td></td>
</tr>
<tr>
<td>Break into parts</td>
<td>Divide and conquer</td>
</tr>
<tr>
<td>Use incremental development</td>
<td></td>
</tr>
<tr>
<td>Model real world</td>
<td>Relate to real world</td>
</tr>
<tr>
<td>Use analogy to real world</td>
<td></td>
</tr>
<tr>
<td>See context/reason/ use for something</td>
<td>Look for the bird’s eye view</td>
</tr>
<tr>
<td>See the larger picture</td>
<td></td>
</tr>
<tr>
<td>See patterns</td>
<td></td>
</tr>
<tr>
<td>See a large system</td>
<td></td>
</tr>
<tr>
<td>Transfer from language to language</td>
<td>Make transfers/connections</td>
</tr>
<tr>
<td>Connect to mathematical formalism</td>
<td></td>
</tr>
<tr>
<td>Relate to something already learned</td>
<td></td>
</tr>
<tr>
<td>Relate different levels of abstraction</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Abstract/understand stuff

In contrast, some subjects discussed learning and getting unstuck at a higher level. Many quotes relate concepts to real world examples, relate pointers to a television remote control or a leash, relate objects to different rooms in a house. And, while Subject1 talks about examples, it is the idea of putting it in context that stands out:

But to see all of those different examples—that same idea in all those different contexts and to figure that out on my own really just taught me like critic-like thinking skills.

Several subjects talk about breaking problems down into smaller parts, and Subject10 discusses the upward view, the larger picture in object-oriented design:

Once you do it enough you stop thinking about that and you think about it in a bigger view if that makes sense.

Furthermore, Subject4 addresses the importance of seeing patterns:

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.

And, Subject1 demonstrates the importance of making connections and formalism:

And it wasn’t until I took really functional programming after discrete math that I realized solutions could be simpler if I used recursion, less lines or just easier to code or easier to reason about.

3.4 “Use the Force”

A number of strategies involve the students using their willpower or character: telling themselves to remember things, to be more persistent, or to sit and think. For example, the following quote from Subject4 illustrates the value of remembering things:

And then you just, you know, you get that little nugget in your head and you carry that on to the next time. So, the next time you have something
with a pointer that isn’t working, you go, Okay, I need to do this. And if it still isn’t working, then it’s something else.

Subject11 talks about the value of being persistent:

And just by just staring at it and continually like trying to grasp it, I eventually got like a small piece of it and understood it.

These are generalized strategies that can be applied on top of other specific strategies. The name reflects the aspects pertaining to personal characteristics, an allusion to the movie Star Wars, where the main character is admonished to “Use the Force”—to trust his intuition or inner strength.

4. RELATED WORK

Perkins [12] discusses three different sorts of learners—active, social, and creative—and how they might respond to different forms of constructivist learning. These groups match up fairly well with our abstract strategies: Concrete/do stuff for the active learners, Inputs/interaction for the social learners, and Abstract/understand stuff for the creative learners. Work on “learning styles” [5] indicates that different students may prefer different strategies; for example, visual learners are more likely to draw diagrams, and verbal learners, to read. In any case, our data suggest that successful students often apply multiple strategies to learning a given concept. McKeachie et al. [10] found that the teaching of learning strategies in a course provided benefits to students in their other courses.

Biggs [1] stresses the importance of student activities (as opposed to instructor activities) in learning. Ramsden ([14], p.155) agrees: “Passivity and dependence on the teacher […] provide an excellent basis for surface approaches” and “deep approaches are associated with activity and responsibility in learning.” The strategies seen used by these students are nearly all centered on student activities: even the Learn from other people strategies tend to be driven by student actions, such as seeking out peers or others to learn from. Trigwell et al. [16] discuss the role of “students’ intentions associated with their strategies”; the students interviewed here seem to have been motivated to learn, at least in their retrospective explanations.

A number of studies consider the relative value of deep vs. surface approaches to learning, see e.g. [9], generally with the assumption that deep is inherently better. This view is not exactly supported by work of Hughes and Peiris [7], who investigated students’ approaches to learning programming in terms of deep, surface or strategic approaches—the strategic approach being to “plan their work according to an awareness of tutor’s expectations.” The results showed that course performance had a strong negative correlation with taking a surface approach, no strong correlation with taking a deep approach, but a strong positive correlation with using a strategic approach. Further, Kember and Gow [8] found cultural differences in the relative effectiveness of surface and deep approaches. We saw a range of successful strategies from surface (Practice/drill) to deep (Connect to mathematical formalism); we did not see Hughes and Peiris’s strategic approaches, but we did not discuss general motivation in our interviews.

Debugging is one area of getting unstuck which has been studied in depth. In particular, Vessey [17] identified a hierarchy of goals used by programmers while debugging. Novices stated more hypotheses than experts and tended to stick with their hypotheses so that they failed to understand the program structure.

In addition to work on learning strategies, there has been a good deal of work on student strategies used in programming, program comprehension, and problem solving.

Robins et al. [15] identify lack of programming strategies as a cause for problems for novice programmers. Davies [3] reviewed “studies that have addressed the strategic aspects of programming skill”, and suggests that “the strategic elements of programming skill may, in some case, be of greater significance than the knowledge-based components.”

Fitzgerald et al. [6] identified 19 strategies used by novice programming students when solving examination questions that involved reading and understanding code. They found that success was determined both by the strategies chosen and how well they were employed.

There is quite a large literature on problem-solving strategies, most notably Pólya [13]. Wankat and Oreovicz ([18], chapter 5) summarize a large numbers of studies comparing novice and expert problem solvers. Although problem solving and concept learning are quite different, we see some similarities. The stated expert responses to being stuck are given as “Use heuristics”, “Persevere”, and “Brainstorm.” While the first of these was not seen in our data (except perhaps Use analogy to real world), the others are seen as Be persistent/don’t stop and Discuss.

5. DISCUSSION

None of the strategies the students mentioned was surprising in itself. But together they had a surprisingly long—and thoughtful—list of suggestions. Much of what they suggested was social: several of the students pointed out that it was with the help of their friends, or by asking their instructor or another faculty member, that they actually learned. They mentioned how important it is to relate new knowledge to something you already know. They said that they like step-by-step instructions when learning something new. They stressed the value of persistence and practice, even inventing tasks above and beyond the assigned work.

Some immediate implications for teaching are that we can give the students what they (or some of them) find helpful: the opportunity to discuss their work with their peers, good examples that relate to what they already know, step-by-step instructions, optional extra tasks, etc.

But experienced instructors already do these things. This study provides a deeper understanding of student experience, and a list of possible ways to get unstuck. Just as we might teach debugging—what to do when your program doesn’t work—we can also explicitly discuss how to debug your own learning.

6. CONCLUSIONS AND FUTURE WORK

We have investigated graduating students’ successful strategies for getting unstuck when learning new concepts. All but one of our students recalled getting stuck at one point or another—it is unlikely that a student makes it through a computing degree program without this experience! But the successful students, it seems, have strategies for learning,
Although they get stuck, they find a way to get unstuck.

Altogether we found 35 strategies which we have categorized into groups. The result shows that all students used several strategies, and that the strategies are surprisingly diverse. The importance of social interaction, and the active responsibility taken by the students are striking. This is consistent with previous research emphasizing the importance of students taking responsibility for their own learning. Many strategies discussed by the students can be used either with a surface or a deep approach. The data show that our successful students have used the strategies with a deep approach, actively striving for learning.

It may seem obvious but we think it is worth pointing out that the students experienced and overcame their problems in different ways. It is important for us as teachers to not only acknowledge this but also plan our courses with this in mind. We do not think there is a “one-size-fits-all” recipe for how to learn computing concepts; instead teachers should be aware of strategies students use and regard as successful, and be prepared both to encourage the students to use the strategies they prefer, and to help them to learn new ones.

The results so far suggest a number of areas for future study:

- Learning strategies that students use when they are not stuck. Are other successful strategies used in more “routine” situations that were not considered in this study?
- Learning strategies used by unsuccessful students—either strategies used unsuccessfully, or by students who fail to complete their degrees. Do these students use different strategies? Do they attempt to use the same strategies without success?
- The use of strategies by students at various points in their degree programs. Do the use of strategies develop as students gain experience, or do they gain experience using strategies they always knew, or a mix?
- Applicability of strategies. Are there correlations between the strategies used and either particular concepts being learned, or the preferred learning style of the student?

We think that a better understanding of the learning strategies that students use (and could be taught to use) could lead to improved learning and teaching. We (the authors) have found that what we know so far has already affected how we teach in a positive way.

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 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.

7. REFERENCES


