Harnessing Surprise: Tales from Students’ Transformational Biographies

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

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

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

ABSTRACT
Transformational biographies are used to shed light on situations when students have experienced ‘surprise.’ These biographies are examined in light of a philosophical position that surprise is useful in engendering learning, and, that as educators, we should try and set up situations in which our students experience it more often. The paper reports on a grounded-theory inspired examination of 108 students’ transformational biographies for evidence of surprise. Students clearly perceived an ‘anomaly or contradiction to a previous belief or understanding’ in just under half of them. The paper then goes on to further identify different kinds of surprise and the triggers that caused them and concludes with suggestions for enhancing learning by capitalizing on students’ surprise.

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

General Terms
Human Factors.

Keywords
Surprise, Student Biographies, Attitudinal factors, Threshold Concepts, Transformation, Work Experience.

1. INTRODUCTION
Do students learn better if they experience ‘surprise’ during their learning process? And if so, how can we, as educators, use surprise in our teaching?

In this empirically based work, we examined written biographies collected from students in the second half of their degree programs in Sweden, Wales, and the United States. The students were asked about “transformational learning experiences”, and we were interested to note that many of the experiences that they described involve a clear element of surprise. This caused us to specifically analyze the biographies for further evidence of surprise, and to dig deeper in order to determine how student surprise can be harnessed for the purpose of enhancing the learning experience. A theoretical basis for this work is discussed in Section 2. The methodology is described in Section 3 and our findings are reported in Section 4. If we accept that surprise is a useful learning tool, then what caused it in our students? This is examined further in Section 5. Section 6 comprises a discussion of how instructors can use these results to help students learn computing concepts inside and outside the classroom.

2. BACKGROUND
This research was initially motivated by a search for threshold concepts in computing. In this section, we begin by discussing how surprise might be incorporated into theories of learning, and then go on to relate this to threshold concepts.

Lucas [6] discusses surprise in the context of critical reflection in accounting education, and points to moments of surprise as involving:

… perceptions of anomalies or contradictions … This involves discrepancies between one’s expectations and one’s perceptions of an experience and a trigger event is needed. Secondly, there is a stage of appraisal. This may involve, on the one hand, a minimisation or denial or, on the other, an opening up to unknown possibilities. The third stage involves exploration. This might involve the development of new identities and role models, a public declaration of a change of state, and the testing of new ways of thinking and acting. The fourth stage requires the development of alternative perspectives.

Adler, in a philosophical paper about the use of surprise in education, defines surprise as occurring when one “recognizes an occurrence or event that is contrary to one’s expectations.” [1] The response to surprise that encourages learning is that one’s reasoning was faulty but ‘correctable.’ This then provides intrinsic motivation to learning, and also helps the learner to focus their attention on important concepts.

How can such important concepts be identified? Meyer and Land [8] have proposed using threshold concepts as a way of characterizing particular concepts that might be used to organize the learning process. Threshold concepts are a subset of the core concepts within a discipline, and are characterized as being:

- **Transformative**: they change the way a student looks at things in the discipline
- **Integrative**: they tie together concepts in ways that were previously unknown to the student
- **Irreversible**: they are difficult for the student to unlearn
- **Potentially troublesome** (as in Perkins [10]) for students: they are conceptually difficult, alien, and/or counter-intuitive
• Often boundary markers: they indicate the limits of a conceptual area or the discipline itself

In previous work, it has been determined that although the idea of threshold concepts may be quite appealing, these concepts are less easy to pinpoint in computing than one might assume. A number of potential threshold concepts have been discussed, however, particularly: object-orientation and pointers/memory management in [2] and abstraction in [9]. In addition, students’ experience of learning these concepts has been characterized as the ‘Liminal Space’, which corresponds to some of the aspects Lucas refers to above, particularly those relating to identity and change of state [3].

There seems to be synergy here with the idea of surprise. When Adler says that surprise helps the learner focus on ‘important concepts’ perhaps where we see surprise, we are also seeing potential threshold concepts.

Recent advances in neurology, and the use of MRI scans, have provided researchers with more evidence that at least memory and the learning of concepts is enhanced by surprise and that this has implications for teaching and learning [4]. In terms of harnessing the power of surprise in teaching and learning we may also make reference to the ideas of variation theory, as developed by Marton and others [6]. In variation theory, learning is understood as developing richer ways to see phenomena in the world. This involves seeing new features of phenomena, not discerned before. Variation theory emphasizes variation and discernment as key words in this process. A necessary, but not always sufficient condition for discerning new features is that the student has the opportunity to experience variation in the learning situation in a way that highlights these specific features.

Variation theory emphasizes that it is possible to create learning conditions that enable students to discern new features of phenomena by varying some things and keeping others invariant. Marton et al. [7 p.16–17] discuss so called patterns of variation. These patterns describe different combinations of systematic variation and invariance. One of the patterns, the contrast pattern, is explained by Thuné and Eckerdal [11] as contrasting “a phenomenon P to other related phenomena, to make it possible to discern P as a phenomenon distinct from other phenomena.”

Surprise can appear when students experience a distinct contrast. Nearly all of the biographies were collected as a regular assignment associated with a course taken by computing majors in the second half of their computing degree programs. The students at Institution A, although in the second half of their program had only completed two programming courses; they were generally older, however, and often had industry experience. These biographies, since they were collected from a Technical Communication course, had been reviewed by the instructor for style only and were then revised by the student.

The data led the authors to consider the relationship between learning and surprise. Many students either used the words ‘surprise’ or ‘amaze’, or variations on these terms, or else their biography indicated surprise in some other way.

The emergence of theories from the data is part of a grounded theory approach that originates in the work of Glaser and Strauss [6], who outline grounded theory as a way to “[discover] theory from data systematically obtained from social research” (p. 2). Theories may emerge organically from the data through a process of collection, coding and analysis of data. In this sense, this paper then uses a grounded theory approach to discover a preliminary theory that links surprise and learning in computing.

Biographies that used obvious words were easily pulled from the others for further analysis. In addition, all the biographies were examined by at least two researchers looking for other examples of manifestations of surprise. For the latter biographies, when the rare conflict arose, two researchers worked together to consensus.

The resulting 52 biographies (30 using a keyword and 22 with other indicators) were then examined more closely to determine in what way surprise was a factor identified by the writer. While this work has some quantitative aspects, it primarily uses qualitative research methods, specifically, a thematic approach.

3. METHODOLOGY

Data were collected as written biographies from computing students. Students were asked to identify and describe a computing concept that transformed the way they see and experience computing. Included in the instructions were two sample transformation biographies: the first talked about writing long, repetitive code, and learning about functional decomposition; the second described learning about complexity analysis. A copy of these instructions may be found in [12].

These data were collected from 108 students from 5 institutions in Sweden, the United Kingdom, and the United States. For analysis, the student biographies, where necessary, were translated into English by the researcher. Biographies were collected in a Technical Communication class at Institution A in the U.S.A. (62), in an Agile Methodologies class in the U.K. (35), in an Ethics class at Institution B in the U.S.A. (8), and from students in Sweden (3) who were not associated with a particular class.

<table>
<thead>
<tr>
<th>Table 1: Number of biographies exhibiting surprise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Word Used</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Surprise</td>
</tr>
<tr>
<td>Opened my eyes</td>
</tr>
<tr>
<td>Amaze</td>
</tr>
<tr>
<td>Epiphany</td>
</tr>
<tr>
<td>Others (awe, unexpected)</td>
</tr>
<tr>
<td><strong>Total number of biographies with words (some had more than one word)</strong></td>
</tr>
<tr>
<td><strong>Number of biographies identified by ‘impression’ (without a specific word)</strong></td>
</tr>
</tbody>
</table>
Object-orientation and related concepts, such as inheritance, trees, the halting problem, and TCP/IP, discussed in one biography, for example, security, binary search appeared more than once, but the majority of concepts were only of surprise, a number of concepts were found. Several concepts with a concept sometimes the surprise that students evinced was associated not 4.2 with student's own performance

To perform in a given situation.

Surprise associated with a concept

The students were asked to identify computing concepts that transformed the way they see and experience computing. In investigating what concepts were related to students' experience of surprise, a number of concepts were found. Several concepts appeared more than once, but the majority of concepts were only discussed in one biography, for example, security, binary search trees, the halting problem, and TCP/IP.

Object-orientation and concepts related to it, such as inheritance, polymorphism, encapsulation and object were by far the most common concepts mentioned. About 20 biographies discuss one or more of them.

One example where surprise is related to object-orientation is when subject 21 experienced a positive surprise when finally understanding the idea of modeling in object-orientation. This left a lasting impression:

521 - My code became very long and hard to understand as just a long list of commands. After realising that this was not the right way to go about making the program I thought back to one of the first lectures I was given on Object Orientation. It then dawned on me that the classes were representing real world objects, the dog class could be thought of as an actual dog, with a name and colour represented as attributes in the class. This was a major breakthrough in the development of that worksheet.

Similarly, subject 50 discusses inheritance:

550 - While working on the assignment I was amazed by how much less code I had to write to complete the same programming project.

With this new insight the student shows appreciation:

550 - If only I had this knowledge at the beginning of the course, who knows what kind of headaches inheritance might have saved me.

4.2 Surprise associated with behavior

Sometimes the surprise that students evinced was associated not with a concept per se, but with something else such as a technique, or a modification of behavior.

Biographies that mentioned a technique usually referred to it in terms of surprise at its efficacy. Students had been introduced to the technique before but that they had never seen the value of it. Designing before coding and elegance in coding were very common here:

513 - This really opened my eyes to the major difference that uni was going to make on the way I coded, ... I began to see the importance of fore-thought when writing a program rather than just diving in _head-first_ and coding something that in the end could be completely replaced by something much easier and more efficient.

It is heartening to see students eventually grasping this kind of technique. Other techniques that caused students surprise included running queries through a query analyzer, including good source code comments, and using Open Source Software:

579 - It was almost 300 times faster than my previous query. The performance increased exponentially and I was enchanted by the improvement I made.

593 - The uncommented four page code took me twelve straight hours to figure out what the program was trying to do, and how to make it compile correctly. In contrast, the commented six page source code took me two hours.

585 - For years, I happily ran their inefficient, proprietary, closed-source software, oblivious to alternatives...my eyes were opened to alternatives, and I began to experiment.

4.3 Surprise with student’s own performance

Some students were surprised that programming was easier than they had expected, for example:

58 - I really did surprise myself though, and I learned enough in just a few hours to create the framework of a working interactive website with secure login.

Others discovered that they were not as good as they had thought:

569 - Within the first couple days I realized I was in way over my head. I had spent my time on a Windows machine playing video games, and thought I knew everything because I could update a video card driver. ... I knew so little I was pretty sure I was going to be fired. This was surprising and scary for me, because I came into the job thinking I was settling for a position I would quickly get bored with.

The students quoted above continued on in Computer Science with success. The negative experience reported by 569 caused him to reevaluate his preparation for the job: “It has been almost two years since I started and I have absorbed as much knowledge as possible in that time.”

A few students mentioned Internet or gaming communities and the power of social involvement in drawing them into computing:

557 - I decided to go try out this new concept called “chat rooms” and when I entered there were hundreds of people chatting away about anything and everything... This form of socialization was very new and exciting and it was the beginning of my passion for computers and internet.

One student, however, reported a different result from his surprise experience, one that made him think about his career:
S107 - Before this experience, I thought I was never ‘bad’ at anything. In my experience in academics I was never truly ‘stuck’ at any assignments, or projects. … because of that, I was never forced to find something that I was good at doing. However, this experience of failing to implement a complete program … made me think about what I actually wanted to do for my career.

5. TRIGGERS FOR SURPRISE

We investigated where students were and what they were doing at the time of their surprise. The list of these triggers is not unexpected:

- Instructors
- Comparing to previously-learned material
- Working on a project or other work
- Peer pressure
- At a job
- Something external (not job or school)

But digging deeper gives us insight into the triggers for their surprise. Instructors make an impression on students. Some were quoted, e.g., “There’s no point in reinventing the wheel all over again” in the context of inheritance. After pointing out that an instructor showed the comparison of a O(n) vs. O(log n) algorithm, a student relays surprised delight: “I had no idea that such a process could be optimized to such a great extent.” Other instructors made an impression as though they were personally responsible: “Then, to my relief, my professor introduced me to the concept of model-view-controller […].” Moreover, professors can be a catalyst for students’ imagination:

S92 - Earlier this year, I was sitting in […] trying hard to listen to the professor’s dissertation on the fundamentals of C++ pointers. As I listened, I began to visualize the world of computing differently. … In my mind, I could see a massive, interconnected web of arrows pointing around a great room with many boxes. The boxes were of all different sizes and shapes -- each with its own functionality.

While most of the triggers that directly came from instructors are seen positively, there is often a fine line. One student clearly relays this: “Following the instructor made it easy to do the assignment, but I really didn’t know what I was doing at first.”

Students displayed surprise with relation to previous things they had learned. Many students who had learned procedural programming first, commented positively on object-oriented programming: “This new concept was completely unlike the disorganization of my previous efforts epitomized.” and “Sure, object-oriented design had been used for several years before I discovered it, but for me, it was as if I had just invented the wheel or something. Eureka!”

One student had a major revelation when learning about the architecture of processors and how logical gates can be arranged to create complex circuitry:

S43 - I had always been confused by how the programs I wrote could ultimately be reduced to nothing but electrical signals, but I suddenly saw how you could create a simple adder.

Another student, upon seeing the proof of the Halting problem’s undecidability, said it “changed my view of computers and their capabilities.”

Projects were often catalysts. An assignment to simulate a traffic intersection and another to simulate a lunar lander helped students understand the interaction of objects in object-oriented programming. Similarly, a student found the modularity of a Sudoku problem-solving program “amazing” and a group project solving a Trivial Pursuit team quiz application highlighted the importance of object-oriented concepts.

Working on recursion, one student describes how writing out a visual representation helped: “Writing the function calls and blocks of memory allocated out on paper finally allowed me to understand.”

Peer pressure was a trigger. A student describes how a classmate “was horrified” upon seeing the student’s finished code and says “after seeing my classmate’s expression and his code, I was truly embarrassed […].” Another student came to fully appreciate software design practices after tutoring a peer. Previously, he “lazily” completed the required design documents after the coding, but came to understand the purpose of designing and when “the design process was taken seriously,” less time was spent on the assignments.

Triggers also occurred through a job. One student used inheritance to enhance the security of their web-based application. Another realized “I had no experience with important website applications like IIS, Apache, or MySQL.” Another student discovered artists who were incorporating computer programs into their artwork and realized that “artistic creativity also exists in computer programming.”

Lastly, triggers could be external to anything that happened at school or work. For one student a network crash forced a learning experience that eventually led to being “reborn” with the knowledge of how networks communicated. Another student was “excited” to do “surfing on the Internet Highway.” Still another had an “epiphany” while working on a hobby project to convert an old program over to C++, realizing inheritance and polymorphism could be used to easily manipulate data.

6. DISCUSSION

To some extent surprise is serendipitous. One student noted this himself “Was it my randomness that one class had become my stepping stone in my life?” But, as educators, how can we harness it in our teaching and curriculum design?

A surprising suggestion from the new literature on brain activity is to begin lectures with something novel “…novelty seems to promote memory. This finding gives teachers a potential tool for structuring their lessons more effectively. Although most teachers start a lesson by going over material from the previous class before moving on to new subject matter, they should probably do just the opposite: start with surprising new information and then review the older material.”[4] There is some evidence of this in the biographies. Students noted when something that an instructor did “opened their eyes”; it would be interesting to know at what point in a lecture this occurred.
Some things instructors already know and try to do – incorporate meaningful challenging assignments in our courses. Can we do more? Students react differently to different projects. While some students react to games, others prefer practical applications, so the best compromise is to vary the assignments at an attempt to appeal to all students.

As faculty, we also know how much is learned by teaching, by the action of explaining out loud to someone else. In the biographies students relay gaining much from peers in different ways. Can this be productively channeled? Group projects facilitate peers learning from each other. Having to discuss problems forces students to discuss issues out loud that they might otherwise only mull over in their heads. Also, having undergraduate assistants for lower-division courses allows students to tutor.

What peers think about students and their work appears to have an impact and therefore peer interaction can provide a strong motivation to change. However, this is a delicate situation. On the one hand, peer interaction is potentially very effective, but if students may feel embarrassed in these situations. As teachers, we must provide students a comfortable atmosphere where they can trust each other with their work. For example, code could be reviewed anonymously.

The evidence showing surprise for students starting with procedural programming and changing to object-oriented is powerful. This could be interpreted as a reason to teach a fundamentals approach first. Without comparison, students may not understand or appreciate the need for what they perceive as complexities of the object-oriented paradigm. One student describes this sentiment as “Love at first sight” with object-oriented programming. He describes how satisfaction with basic data types did not last and eventually it was difficult to keep track of all the variables. Fortunately, “the most exciting programming class” on object-oriented programming gave him the tools.

The contrast pattern, discussed in Section 2, is a useful tool for introducing surprise [11]. Teachers can use this pattern in a different context to help students discern important aspects of the learning goals that otherwise might be less obvious. In varying some aspects of the experience, students can be led to learn through experiencing surprise. For example, one subject, in discussing inheritance, described an assignment: “The final class assignment was to take a prior programming project and rewrite the program by only using pre-existing classes provided by Sun Microsystems. [...] While working on the assignment I was amazed by how much less code I had to write to complete the same programming project” This contrast helped the student to discern one important idea behind inheritance.

Similar experiences were reported by the students quoted in Section 4.2. One instructor assigned students the problem of debugging two programs, one commented and one without – the uncommented code took six times longer to debug.

There is certainly evidence in this research that what we do as instructors, in introducing new concepts, setting assignments, and supervising work experience and peer tutoring, makes a difference; particularly if we can encourage our students to experience surprise, leading to a desire to correct their faults and hence providing intrinsic motivation to learning.

7. ACKNOWLEDGMENTS
Thanks to our students for allowing us to use their work, to Chris Loftus for collecting biographies, and to Jonas Boustedt, Robert McCartney, Jan Erik Moström, and Kate Sanders for their work on the initial analysis. Thanks also to Uppsala University and to Umeå University for physical and electronic meeting places.

8. REFERENCES