Creative Students – What can we Learn from Them for Teaching Computer Science?

Ralf Romeike
University of Potsdam
Department of Computer Science
A.-Bebel-Str. 89
14482 Potsdam, Germany
romeike@cs.uni-potsdam.de

ABSTRACT
Supporting creativity is a learning objective in general school education. In computer science the creation of products plays an important role. We want to find out the impact of creative action in computer science education classes on task motivation and understanding of computer science concepts. Underlying elements and environments of the creative process are analyzed by interviewing creative students. First results indicate creativity as an important factor for motivation. Furthermore the use of a concept we call "building blocks" helps to achieve creative products.

1. INTRODUCTION
Creativity is a phenomenon of human behavior, which is generally valued, admired and desired from politics and industry. With that the encouragement of creativity found its way into schools and curricula. It is also aimed for as a superior learning objective in computer science education.

Boden [1] describes two aspects of creative achievements. Historical creativity (H-creativity) describes ideas, which are novel and original in the sense that nobody has had them before. Something that is fundamentally novel to the individual Boden describes as psychologically creative (P-creativity). In an educational context the latter is more interesting. Thus the difference between an exceptionally creative person and a less creative person is not a special ability. It is based on a larger knowledge in a practical and applied form as well as on the will to acquire and use that knowledge.

Empirical studies show that today's professionals criticize the lack of creativity and other social skills in their education. Industry also remarks a lack of creativity in school graduates. School now is trying to fulfill the needs and educate the students more creatively.

We believe that the school subject of computer science can contribute a lot to that issue. Even if the transfer of creativity within different domains is discussed controversially [8], domain specific creativity still should be promoted. Computer science has the potential to keep up with subjects like music, arts or languages, which traditionally are said to enhance creativity. Perhaps it can even perform better in that.

2. CREATIVITY IN CS EDUCATION
Computer science, as computer scientists see it, is a creative field to work in [5]. There is also seen potential for creativity in computer science education. Especially modeling in the context of software development is emphasized as being a creative process [6,9]. It offers students "the possibility to develop creative solutions for problems, to realize and to validate them" [9] and should be revealed as a creative process to the students. It is questionable if this potential is actually applied in models for teaching and in the lessons. Approaches and justifications for computer science education generally tend to focus more on the aspect of problem solving or modeling of a problem with having the implementation of the solution as a motivating bonus at the end.

A lack of awareness for creative student activity is also found in lesson drafts which are presented in journals for computer science education. In an analysis of possible creative student action in 15 lesson drafts only two were explicitly assigning time for that. Sometimes it is even feared that students let their creativity out to much [4]. Obviously noncreative students are easier to handle in class than creative students. This can lead to making creativity even undesired. We think that explicitly planning of creative lesson phases is important. That means instead of only focusing on the recognition of principles of information systems there also needs to be time for designing and shaping them. Divergent learning tasks need to be offered to the learners. The goal of implementation should not be just a testing of paper-found or presented solutions and models, but needs to be implemented actively into the learning process. From lesson observations we assume, that such creative phases can enhance students motivation and raise understanding of computer science concepts. According to Scragg [7], students also have to learn creativity, which comes from insight into the facts, methods, and paradigms of the field. We intend to conduct research on the question if there is interdependency; if creative action conversely will have positive effects to the understanding of content, concepts and their interaction.

3. APPROACH
In our study we ask the question: Does the integration of creative phases in computer science classes enhance students' motivation and which effect do they have on their understanding?

In order to conduct creative lessons and empirically test the outcome we need to find out what creative work in computer science classes is about, especially what it means to students. We need to find out answers to the question: What are the underlying elements and environments of creative working in computer science classes?

Therefore we want to investigate experiences and attitudes of students, who stand out due to creative achievements in the lessons. How is the creative action connected with motivation and interest? Which understanding do these students have of creativity? Which role does creativity play for them? How do they perceive computer science? Which methods are they using?

First hints of possible answers about how to achieve creative products can be found in common literature of psychology and software development. We suggest asking the students in an outline based interview generally about their methods, approaches, strategies and views to computer science. If necessary, methods suggested in literature can be proposed and asked to be evaluated.

4. A PRE-STUDY

In a pre-study for exploring our field of research an outline interview was conducted with a student whose creative achievements stood out in and outside of the lessons. His actions we call creative due to the independent finding and formulation of new ideas (fluency), the change of categories for the subject he worked with (flexibility) and the development of personal novel products, without having a fixed solution at hand before (creative problem solving). During the interview programming came out as the major source of interest in computer science and thus was the main topic in the interview.

Practical work, in particular programming, was considered to be more interesting to the student. In working practically he enjoys the feeling of being able to be creative. With little effort he can accomplish a lot. The main source of motivation comes out of an aiming for self-realization. This is perceived as creativity and results in a final product.

For the student programming is a creative task. He states that it is creative, when a personal method is used for producing something. This understanding of creativity relates to the definitions of P-creativity and problem solving. To problem solving it adds the aspect that there is an idea about how a result may look like, but it is basically just a direction and not concretely defined. Thus the discovery of the direction, different possibilities and the uncertainty about the result play an interest raising role. In comparison with other school subjects, he argues that computer science is as creative as music and arts, even if this is not generally perceived the same by the majority of other students.

Constructs for programming are perceived as “building blocks”. Utilizing these building blocks programming appears to be easy. In this sense programming means putting blocks together as putting together bricks when building a house. In this case the bricks are fundamental programming constructs, such as variables, commands, iteration and so on. The principles and concepts of programming are applied when putting these bricks together.

The construction kit concept is well known in computer science: Data types are modelled using the building block principle; for the modelling of processes the different programming paradigms with their construction kits are used. Another construction kit would be e.g. the catalogue of fundamental ideas, which describe important concepts and strategies.

Similar results were found by Bruce et al [2] when they observed that students use building blocks in a programming course as an approach to gain understanding of the concepts as well as an approach for writing programs.

In contrast to the structured approach to programming in many didactic concepts the student often claims not to follow a certain strategy. More important is the possibility to try out things and proceed intuitively. The compiler permanently guarantees feedback which influences the problem solving process positively. The space of possible solutions in this way can be tested and the knowledge about it is extended. Glass makes a similar point about the use of “selective trial and error” in software development [3].

5. CONCLUSIONS

The majority of concepts and theories about how to teach computer science in school are based on deductive approaches. Starting from the computer science perspective, especially software development, promising content and concepts are carried over and adapted to the classroom. In contrast our approach starts bottom-up. Applying the experiences of creative students a catalogue of hints for creative working shall be developed. Promising first results suggest motivation and understanding are closely related to creative action and vice versa. In teaching concepts this aspect does not find enough attention. In continuing research on that issue we try to provide a basis to allow students to harness their creativity.

To receive a more reliable collection of elements supporting creative action more interviews need to be done. Therefore the requirements for selecting creative students need to be concretized, a variety of students need to be chosen and the interview outline designed.

Finally the findings need to be applied in an empirical study to draw conclusions about the effect of creative phases in the lessons.

6. REFERENCES

[1] Boden, M. A. *The creative mind: myths & mechanisms*. Basic Books, London, 1990.

[2] Bruce, C., Buckingham, L., Hynd, J., McMahon, C., Roggenkamp, M. and Stoodly, I. Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. *Journal of Information Technology Education*, 3:143-160, 2004.

[3] Glass, R. L. *Software Creativity*. Prentice Hall PTR, Englewood Cliffs, 1995.

[4] Janneck, M. *Partizipative Systementwicklung im Informatikunterricht.* LOG IN 138/139, LOG-IN-Verlag, Berlin, 2006, 60-66.

[5] Leach, R. J. and Caprice A. A. The Psychology of Invention in Computer Science. In *Proceedings of the Psychology of Programming Interest Group Workshop.* Brighton, UK, 2005.

[6] Schulte, C. *Lehr-Lernprozesse im Informatik-Anfangsunterricht : theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II*. Diss. Univ. Paderborn, 2003.

[7] Scragg, G., Baldwin, D. and Koomen, H. Computer science needs an insight-based curriculum. *ACM SIGCSE Bull*., 26(1):150-154, 1994.

[8] Sternberg, R. J. *Creativity: From potential to realization*. APA, Washington, DC, 2004.

[9] Thomas, M. *Informatische Modellbildung: Modellieren von Modellen als zentrales Element der Informatik für den allgemeinbildenden Schulunterricht*. Diss. Univ. Potsdam, 2002.