Abstraction Power in Computer Science Education

Jens Bennedsen
IT-University West
Fuglsangs Allé 20
DK-8210 Aarhus V
Denmark
Email: jbb@it-vest.dk
Michael Caspersen
Department of Computer Science
University of Aarhus
Aabogade 34
DK-8200 Aarhus N.
Denmark
Email: mec@daimi.au.dk

Introduction

A substantial amount of research has been conducted in order to identify variables that are predictors of success of students aiming for a university degree. Investigated variables encompass among other things gender (Bennedsen & Caspersen, 2005; Rountree, Rountree & Robins, 2002), the educational level of parents (Ting & Robinson, 1998) and ACT/SAT scores (Bennedsen & Caspersen, 2005; Rountree, Rountree & Robins, 2002; Sanders, 1998). These factors represent scientific competences (math score for example) or unbiased factors (e.g. gender). However, these variables do not account for all of the variation in academic success. Many teachers within computer science seem to find abstraction power to be a major success factor (see e.g. Alphonce & Ventura, 2002; Nguyen & Wong, 2001; Or-Bach & Lavy, 2004), but to our knowledge no one have done research to verify if abstraction power is actually a predictor for success. We believe that one of the reasons for this lacking investigation is the lack of a suitable definition of levels of abstraction and test-instruments for evaluating the students' level of abstraction.

Research on success factors has been conducted both in the general context of education, within computer science, and in the more topic specific area of introductory programming (Bennedsen & Caspersen, 2005; Bergin & Reilly 2005; Byrne & Lyons 2001; Leeper & Silver 1982). Even in the area of introductory object-oriented programming there has been research trying to establish general factors to predict success or failure of particular students. Especially the work of Phil Ventura (2003) focus on a systematic evaluation of hypothesis related to the factors for success of an introductory programming course using an objects-first approach (The Joint Task Force on Computing Curricula 2001). The results are documented in (Ventura 2003; Ventura & Ramamurthy 2004).

As always, there are some preconditions to the research. One important precondition is the characteristics of the course that founds the basis of the research. Ventura used a CS1 course with a graphics early approach (Ventura & Ramamurthy 2004, p. 241). In our research, we look for potential success factors for an introductory programming using a different approach than Ventura's, a so-called model-based approach to programming (Bennedsen & Caspersen 2004). Both courses are objects-first.

Our hypothesis is "A person's abstraction power has a positive influence on his or her ability to program"

Abstraction power

Many educators within the computer science field argue that abstraction is a core competence – see e.g. (Alphonce & Ventura, 2002; Nguyen & Wong, 2001; Or-Bach & Lavy, 2004). However, no one has defined what is meant by abstraction. In this research, we use the levels of cognitive development and test defined by Shayer and Adey (Adey & Shayer 1994; Shayer & Adey 1981) as a way to define and describe abstraction power. Based on Piaget's work on the nature of knowledge Shayer and Adey define eight stages of cognitive development of pupils (Adey & Shayer 1994 p. 30)

1	Pre-operational
2A	Early concrete
2A/2B	Mid concrete
2B	Late concrete
2B*	Concrete generalization
3A	Early formal
3A/3B	Mature formal

3B	Formal generalization

Table 1: Cognitive development stages

Shayer and Adey use it in the age range of 5 to 16 year old pupils. We use the stages on students in the range of 18 to 22. Shayer and Adey found that at the age of sixteen 30 percent of the pupils were at stage 3A and only approximately 10 percent at stage 3B. Further more they found that the curve describing the progression of stages was very flat at that age. We therefore believe it will be relevant to use this stage model to describe the students cognitive development stage.

Based on Inhelder and Piaget (1958), Adey and Shayer describe what they call "reasoning patterns of formal operations" and group the eight patterns in three groups: handling of variables, relationships between variables and formal methods. See Adey and Shayer (1994) p. 17-25 for a more exhaustive description. A person can of course be at a higher development stage in one of these reasoning patterns, but "one would not find an individual competently fluent with one or two of the reasoning patterns who would not, with very little experience, become fluent with them all" (Ibid. p. 17).

Shayer and Adey have developed several tests to determine the students' cognitive stage. These tests focus on several of the reasoning patterns, but since the students "with very little experience, become fluent with them all" we find it sufficient to use only one test. We will use the so called "pendulum" test; a test that has been used for a long time to test children's understanding of the laws of the physical world (Bond, 2004). Shayer and Adey argues that the pendulum test is particular focused on testing the cognitive development stages from 2B to 3B (Adey & Shayer, 1994, p. 30) – the span of cognitive stages we find relevant to test for our age group. It furthermore focuses on one of the most relevant group of reasoning patterns for computer science education: handling of variables.

The ability to program

The term "learn to program" is not a well-defined term – there are many interpretations of this. du Boulay (1989, p. 283f) describes five overlapping domains and potential sources of problems that the learner must grasp in order to learn programming:

1. General orientation: What is the general idea of programs, what are they for and what can be done with them?
2. The notational machine: An abstract model of the machine when it executes programs (i.e. the meaning of the running program).
3. Notation: The syntax and semantics of the programming language used.
4. Structures: (Abstract) solutions to standard problems, a structured set of related knowledge.
5. Pragmatics: The skills of planning, developing, testing, debugging and so on.

We will use du Boulay's five domains as a guide for the content the students must learn. However du Bouley does not describe the level of detail that each domain must be learned. In this study we will study students who have participated in two programming courses at university level – an introductory course (a CS1 course called dIntProg) and a more advanced course (a CS2 course called dProg2). These courses have their particular interpretation of what it means "to learn to program", but both of them focus more or less on all of du Boulay's domains.

The context of the study

The students in this research all study at the faculty of science at the University of Aarhus, Denmark. They all follow dIntProg as an obligatory part of their study program. The course runs for seven weeks. One to two weeks after the course there is a lab test with a binary

pass/fail grading. Every week there are four lecture hours, two lab hours and two class hours with a teaching assistant (TA). Besides scheduled hours, the students are supposed to work approximately seven hours per week in study groups or on their own.

There are roughly 300 students from a variety of study programmes, e.g. computer science, mathematics, geology, nano science, economy, multimedia, etc. 40 % are majors in computer science, and they are the only group of students that continue with the second half of CS1. The rest of the students proceed to other programming courses related to their fields (e.g. multimedia programming, scientific computing, etc.).

The goal is that the student learns the foundation for systematic construction of simple programs and through this obtains knowledge about the role of conceptual modelling in object-oriented programming. Furthermore, it is the goal that the student becomes familiar with a modern programming language, fundamental programming language concepts, and selected class libraries.

The course content is fundamental programming language concepts, object-orientation, and techniques for systematic construction of simple programs. For further details on the structure and contents of the course, see Bennedsen and Caspersen 2004 and Fjuk, Berge, Bennedsen and Caspersen (2004).

The students who will major in computer science follow the second programming course (dProg2). The content of this course is (dProg2, 2005)

Advanced programming language concepts and techniques for design, specification and implementation of slightly larger programs.

- Language concepts: Abstract data types, polymorphism, events, exceptions, streams and threads.
- Design: General design criteria and selected design patterns.
- Specification and implementation: Separation of specification and implementation, interfaces, pre- and post-conditions, simple invariance techniques, sweep and iterators, design of simple class hierarchies, abstract classes, simple recursive data structures, application of standard frameworks (particularly graphical user interfaces).

Data

Several different data sources will be used in this study. Information comes from the administrative system at the university (major), the final exam (the score in the exam), and an experiment conducted by the authors (their abstraction power)

Success. The final exam is a practical, programming test. The official result of the exam is a binary grading (pass or fail). In order for this research to be able to analyse the results at a finer grain, we will post-mark all the students' solutions. The result of the more fine-grained marking is a ten-ary grading on the scale (0, 3, 5, 6, 7, 8, 9, 10, 11, and 13) (see Exam score, 2005).

In order to pass an exam, a student needs a grade of 6 or more. To validate the results of the post-marking, the post-marking is compared to the official results of the exam in the sense that all the students who passed the exam got a grade of 6 or more and the students who failed the exam got a grade of 5 or less. In order to ensure that the marking was fair, the co-author marked twenty randomly selected answers.

In all the statistical tests, the result of the marking is used as the indicator of success — higher grade means more success.

References

Adey, P and Shayer, M. 1994. Really raising standards: cognitive intervention and academic achievement. Routledge. London, England

Alphonce, C. and Ventura, P. 2002. Object orientation in CS1-CS2 by design. In Proceedings of the 7th Annual Conference on innovation and Technology in Computer Science Education (Aarhus, Denmark, June 24 - 28, 2002). ITiCSE '02. ACM Press, New York, NY, 70-74

Bennedsen, J. & Caspersen, M.E. (2004). Programming in Context – A Model-First Approach to CS1, Proceedings of the thirty-fifth SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia, 2004, pp. 477-481.

Bennedsen, J & Caspersen, M. E. (2005). An Investigation of Potential Success Factors for an Introductory Model-Driven Programming Course. Proceedings of ICER 2005 The First International Computing Education Research Workshop, October 1-2, 2005, Seattle, WA, USA

Bergin, S & Reilly, R (2005) Programming: factors that influence success. SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA. pp. 411-415.

Bond, T. B. (2004). Piaget and the Pendulum. Science and Education, 13, pp. 389 - 399

Boyer, S. P., & Sedlacek, W. E. (1988). Noncognitive predictors of academic success for international students: A longitudinal study. Journal of College Student Development, 29, 218-223.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. Proceedings of the 6th annual conference on Innovation and technology in computer science education, 49-52.

du Bouley, B. (1989). Some difficulties of learning to program. In E. Soloway & J.C. Spohrer (Eds.) *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum.

dprog2. (2005). Retrieved from https://mit.au.dk/da/uddan/udbudbeskriv.cfm?udbudid=2079&lan=en&scope=1&parentelem=2240 December 21, 2005.

Exam score (2005). Retrieved April 30 2005 from http://www.retsinfo.dk/_GETDOCM_/ACCN/B19950051305-REGL (English translation can be found in the bottom)

Fjuk, A., Berge, O., Bennedsen, J. & Caspersen, M. (2004). Learning Object-Orientation through ICT-mediated Apprenticeship. Procedings of the 4th IEEE International Conference on Advanced Learning Technologies, Joensuu, Finland.

Hagan, D., & Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program? ACM SIGCSE Bulletin, 5th annual SIGCSE/SIGCUE conference on Innovation and technology in computer science education, 32(3). pp. 25-28.

Inhelder, B. & Piaget, J. (1958). The Growth of Logical Thinking .Routledge & Kegan Poul. London, England

The Joint Task Force on Computing Curricula (2001). Computing Curricula 2001 (final report), December 2001. Retrieved December 14, 2005 from http://www.computer.org/education/cc2001/final.

Leeper, R. R., & Silver, J. L. (1982). Predicting success in a first programming course. Technical Symposium on Computer Science Education, Proceedings of the thirteenth SIGCSE

technical symposium on Computer science education, Indianapolis, Indiana, United States. pp: 147 – 150.

Nguyen, D. & Wong, S. (2001) OOP in Introductory CS: Better Students Through Abstraction Position paper for OOPSLA 2001, Fifth Workshop on Pedagogies and Tools for Assimilating Object-Oriented Concepts

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

Rountree, N. Rountree, J. and Robins, A (2002).Predictors of success and failure in a CS1 course. SIGCSE Bulletin, vol. 34(4) pp. 121—124.

Shayer, M. and Adey, P (1981). Towards a Science of Science Teaching. Heinemann Educational Publishers, Oxford, England

Ting, S. R., & Robinson, T. L. (1998). First-year academic success: A prediction combining cognitive and psychosocial variables for Caucasian and African American students. Journal of College Student Development, 39, pp. 599-610.

Ventura, P. R. (2003). On the Origins of Programmers: Identifying Predictors of Success for an Objects First CS1", PhD. dissertation, The State University of New York at Buffalo, 2003.

Ventura, P. R. & Ramamurthy, B. (2004). Wanted: CS1 Students. No Experience Required. ACM SIGCSE Bulletin, Proceedings of the 35th SIGCSE technical symposium on Computer science education, Vol. 36(1) pp. 240 – 244.

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. ACM SIGCSE Bulletin , Proceedings of the thirty second SIGCSE technical symposium on Computer Science Education, 33(1), pp. 184-188

Wilson, B.C. (2002). A Study of Factors Promoting Success in Computer Science Including Gender Differences. Journal of Computer Science Education 12 (1-2), pp. 141 – 164