Abstraction Ability as an Indicator of Success for Learning Object-Oriented Programming?

Jens Bennedsen
IT University West
Fuglesangs Allé 20
DK-8210 Aarhus V
Denmark
jbb@it-vest.dk

Michael E. Caspersen
Department of Computer Science
University of Aarhus
Aabogade 34, DK-8200 Aarhus N
Denmark
mec@daimi.au.dk

Abstract

Computer science educators generally agree that abstract thinking is a crucial component for learning computer science in general and programming in particular. We report on a study to confirm the hypothesis that general abstraction ability has a positive impact on programming ability. Abstraction ability is operationalized as stages of cognitive development (for which validated tests exist). Programming ability is operationalized as grade in the final assessment of a model-based objects-first CS1. The validity of the operationalizations is discussed. Surprisingly, our study shows that there is no correlation between stage of cognitive development (abstraction ability) and final grade in CS1 (programming ability). Possible explanations are identified.

Keywords: CS1, success factors, abstraction, model-based programming, objects-first.

1. 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 [4, 17, 24], the educational level of parents [20] and ACT/SAT scores [4, 14]. The variables represent scientific factors (e.g. math score) or unbiased factors (e.g. gender). However, these variables only account for a fraction of the variation of student performance.

Research on success factors has been conducted both in the general context of education, within computer science, and in the more specific area of introductory programming [4, 6, 9, 14]. Also 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 Ventura [21] focus on a systematic evaluation of hypothesis related to success factors of an introductory programming course using an objects-first and graphics early approach [22, p.241]. The results are also documented in [23].

We are specifically interested in abstraction ability as an indicator of success for learning programming. Most computer science teachers find abstract thinking to be a core competence in programming, but to our knowledge no research has been conducted to verify whether abstraction ability is actually a predictor of success of an introductory programming course using an objects-first strategy [3].

2. Abstraction Ability and Programming

Many computer science educators argue that abstraction is a core competence [2, 13, 15, 16, 19].

Nguyen & Wong [15] claim that it is difficult for many students to learn abstract thinking; at the same time they claim abstract thinking to be a crucial component for learning computer science in general and programming in particular. The authors describe an objects-first-with-design-patterns approach to CS1 with a strong focus on abstract thinking and development of the students' abstractive skills.

In [16] the authors argue that abstraction is a fundamental concept in programming in general and in object-oriented programming in particular. The authors describe a three-level ordering of abstraction cognitive activities that the students employ in their solution to a given problem: 1) defining a concrete class, 2) defining an abstract class with attributes only, 3) defining an abstract class also including methods, and 4) defining an abstract class also including abstract methods). An analysis of the students' responses to a test reveals that only 13% apply the highest level of abstraction cognitive activities (level 4) while 65% solve the problem at the lowest level of abstraction cognitive processes. The authors conclude that the major cited advantages of object-orientation are precisely the

same issues that make object-orientation difficult for students.

2.1 Hypothesis

Clearly, abstraction and abstract thinking are fundamental concepts in computer science and key components of learning programming. For programming education (and CS education in general) it is therefore mandatory to explicitly aim at the development of the students' abstractive skills. But furthermore we anticipate general abstractive skills —abstraction ability— to be an indicator of success for learning programming. Our hypothesis is therefore:

> *General abstraction ability has a positive impact on programming ability.*

2.2 Abstraction Ability as Stages of Cognitive Development

To operationalize the first part of our hypothesis we need to define what we mean by abstraction ability and how it can be measured.

Or-Bach & Lavy [16] define abstraction ability in terms of object-oriented programming. However, abstraction ability is a much more general skill often defined as part of the cognitive development stage of a person [11].

Our approximation of abstraction ability is based on Adey & Shayer's theory of cognitive development [1, 18]; this theory is a refinement of Inhelder & Piaget's stage theory [11].

Adey & Shayer define eight stages of cognitive development of pupils [1, p. 30] (table 1).

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*

Adey & Shayer based their stages of cognitive development on a very large research project, CASE, aimed at finding the cognitive development stages of pupils in secondary school [1, p.78 ff]. The research showed a different result than the direct connection between age and development stage originally proposed by Piaget. One of the most important results was that only ~30% of the pupils follow the development expected by Piaget.

Based on [11], 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 [1, pp.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" [1, p.17].

Shayer and Adey have used the eight stages for pupils in the age range of 5 to 16; we intend to use it on students in the age range of 18 to 22. Shayer and Adey found that at the age of 16, 30% of the pupils were at stage 3A and only approximately 10% at stage 3B. Furthermore they found that the curve describing the progression of stages was very flat at that age [1, p.40].

We use Adey & Shayer's stage model of cognitive development to characterize the students' abstraction ability.

To measure abstraction ability defined in this way, we use a reasoning ability test developed by Piaget and refined by Adey & Shayer for testing at the higher end of the stage model.

2.3 Programming Ability as Final Grade in CS1

To operationalize the second part of our hypothesis we need to define what we mean by programming ability and how it can be measured.

In this research we use the results from the final exam of the introductory programming course as an indicator of the students' programming ability. For a more thorough description of the course, see [3].

2.4 A Word on the Operationalization

The hypothesis that general abstraction ability has positive impact on programming ability is operationalized in two steps; abstraction ability is operationalized as cognitive development and programming ability is operationalized as final grade in CS1 as illustrated in figure 1.

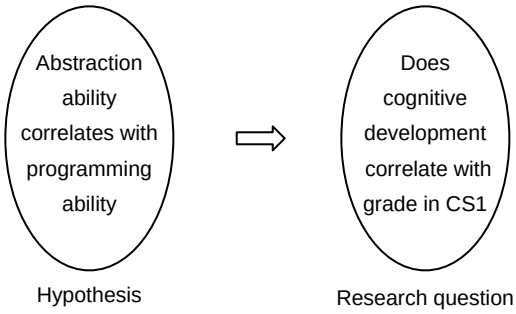

Figure 1: *Operationalization of hypothesis*

Both of these operationalizations are questionable. We discuss this aspect in the section on future work.

3. Research

This section describes the research questions, the data and the statistical analysis used in this work.

3.1 The research questions

Our hypothesis is that there is a positive correlation between the stage of a student's cognitive development (measured as reasoning ability) and the students programming ability (measured as final grade in CS1).

Many reports that math is an indicator of success in programming [4, 9, 14]. Our interpretation of this fact is that it is not specific mathematical competencies (e.g. calculus and algebra) that the students need, but rather the more general notion of abstraction ability required to do math that is needed.

To verify our interpretation, we propose a supplementary research question on the correlation between abstraction ability and mathematical competence. Our two research questions are therefore:

1. *Is there a positive correlation between the stage of cognitive development and the students' results in model-based introductory programming?*
2. *Is math an indicator of the cognitive development stage?*

3.2 The Test

Shayer & Adey have developed several tests to determine the students' cognitive stages. These test focus on several of the reasoning patterns, but because "the students with very little experience, become fluent with them all" we find it sufficient to use only one test. We use the so called "pendulum test"; a test that has been used for a long time to test young persons' understanding of the laws of the physical world [7]. Shayer and Adey argues that the pendulum test is particular focused on testing the cognitive development stages from 2B to 3B [1, p.30], the span of cognitive stages we find relevant for our target group.

The students volunteered to participate in the test. It was given to them in a lecture hall, and they were all informed that the outcome of the test would not be exposed to the lecturer before the exam.

3.3 The Students

The students in this research all study at the Faculty of Science at University of Aarhus in Denmark. They all follow an introductory programming course as a mandatory part of their study programme. The course constitutes the first half of a traditional CS1 course. The course runs for seven weeks. 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. A week after the course there is a practical exam with a binary pass/fail grading. For a more detailed description of the final exam see [5].

In the fall of 2005 there were 263 students from a variety of study programmes, e.g. computer science, mathematics, mathematical economy, multimedia, geology, nano science, etc. Approximately 40 % of the students are enrolled for a major in computer science and they are the only group to 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) if they proceed with programming at all.

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 modeling 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. For further details on the structure and contents of the course see [3].

3.4 Data

Information about the score of final exam comes from the administrative system of the university.

Programming score. The final exam is a practical programming test. The official result of the exam is a binary grading (pass or fail). To allow for a more fine-grained analysis of the results, the students' solutions were post-marked on an A-F scale. To validate the result of the post-marking, the post-marking was compared to the official result of the exam in the sense that all the students who passed the exam got a grade of E or more. Also, the result of the post-marking was checked by a control marking of twenty randomly selected answers. The marking and the control marking agreed.

Math score. The students' math score from high school was used as an indicator of the students' mathematical abilities. The students themselves gave their math score in a questionnaire. A few students did not answer the questionnaire; these students were excluded from the analysis.

3.5 Statistical analysis

We have used a Pearson correlation coefficient test to find if there is a significant correlation between the result of the exam and the cognitive development stage and math score.

Of the 263 students who took the final exam, 145 participated in the pendulum test. They are representative of the overall student group with respect to mathematical skills, gender and intended major.

4. Results

In this section we describe the analysis providing the answers to the two research questions.

4.1 No Correlation Between Cognitive Development and Programming Ability

As described above we have calculated Pearson correlation between cognitive development and programming ability (Table 2). The coefficient, R, is 0.276 which indicates a very weak correlation (a value of at least 0.3 indicates correlation). The significance, P, is less than 0.001.

Pearson correlation test	
R	0.276409
R^2	0.076402
P	0.000764
Observations	145

Table 2: *Correlation between cognitive development and programming ability*

This is a rather unexpected result, since most computer science educators seem to agree that abstraction ability – and thereby cognitive development – is a core competence in programming. Our research cannot demonstrate a correlation between the stage of cognitive development and the students' results in a model-based introductory programming course.

Cafolla [10] reports that the stage of cognitive development accounts for 34 % of variation of the exam score. Cafolla's study is based upon students learning programming in BASIC. It seems unlikely that BASIC programming should require a higher degree of cognitive development than object-oriented programming; we need to investigate this more thoroughly.

4.2 No Correlation between Math and Cognitive Development

We have also calculated Pearson correlation between the score of the programming exam and the math score from high school. The exam in high school is a nation vide test in two parts: a written and an oral test. The written test is administered by the Ministry of Education. We have used the average of the two exam scores as the math score. Of the 143 students participating in the pendulum test, 128 provided their math score.

As can bee seen from table 3, there is hardly any correlation between the students' mathematical ability and their cognitive stage. Again this comes as a surprise as the expected result was a strong correlation between math and formal cognitive development. The result contradicts earlier findings, summarized in [12, p.260].

Pearson correlation test	
R	0.186781261
R^2	0.034887239
P	0.034766
Observations	128

Table 3: *Correlation between stage of cognitive development and mathematical ability*

The correlation that others have found between math and success in programming is not contradicted by our data (R= 0.302191, p=0.000555).

From our experiment we must conclude that math is not just another way of expressing the cognitive development stage and that the correlation between math and success in programming must be related to other aspects of math.

5. Conclusion and Future Work

The result of this study is most surprising. From the outset we were certain that students at a higher stage of cognitive development would get higher scores in the final exam of the introductory programming course. It is not so!

There can be several explanations to this. In this programming course coding is prioritized over design. The cognitive requirements are therefore relatively low, and apparently there are other factors that influence the students' success. We will look into this in future work.

Another potential explanation is the concrete instrument used to assess the cognitive stage: the pendulum test. The pendulum test measures the student's ability to control independent variables in a reasoning task. It could be that this particular competence is not prominent in the course.

Finally, of course, it is questionable to which extend the result of the final exam is a reasonable measure of a student's ability to learn programming.

6. Acknowledgement

It is a pleasure to acknowledge Jens Holbech for his help regarding measurement of cognitive stage (abstraction ability). Also, we would like to thank all the students who volunteered for this study.

7. References

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

[2] Alphonce, C. and Ventura, P. Object Orientation in CS1-CS2 by Design, *Proceedings of the 7th Annual Conference on innovation and Technology in Computer Science Education*, Aarhus, Denmark, 2002, 70-74.

[3] Bennedsen, J. & Caspersen, M.E. Programming in Context – A Model-First Approach to CS1, *Proceedings of the thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, Norfolk, USA, 2004, 477-481.

[4] Bennedsen, J & Caspersen, M. E. An Investigation of Potential Success Factors for an Introductory Model-Driven Programming Course, *Proceedings of ICER 2005 The First International Computing Education Research Workshop*, 2005, Seattle, USA, 155-163.

[5] Bennedsen, J. & Caspersen, M.E. Assessing Process and Product – A Practical Lab Exam for an Introductory Programming Course, Submitted for *36th Annual Frontiers in Education Conference*, San Diego, USA, 2006.

[6] Bergin, S & Reilly, R. Programming: Factors that Influence Success, *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, USA, 2005, 411-415.

[7] Bond, T. B. Piaget and the Pendulum, *Science and Education*, 13, 2004, 389-399.

[8] Boyer, S. P., & Sedlacek, W. E. Non-Cognitive Predictors of Academic Success for International Students: A Longitudinal Study, *Journal of College Student Development*, 29, 1988, 218-223.

[9] Byrne, P., & Lyons, G. The Effect of Student Attributes on Success in Programming, *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, 2001, 49-52.

[10] Cafolla, R. Piagetian Formal Operations and other Cognitive Correlates of Achivement in Computer Programming, *Journal of Educational Technology Systems*, 16(1), 1987-88, 45-55.

[11] Inhelder, B. & Piaget, J. (1955) *De la logique de l'enfant à la logique de l'adolescent: Essai sur la construction des structures opératoires formelles*. Paris: Presses Universitaires de France. Translated by Anne Parsons and Stanley Milgram as *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, New York: Basic Books, 1958.

[12] Iqbal, H.M. and Shayer, M. Accelerating the Development of Formal Thinking in Pakistan Secondary School Students: Achievement Effects and Professional Development Issues, *Journal of Research in Science Teaching*, 37 (3), 2000, 259-274.

[13] Kurtz, B. L. Investigating the Relationship Between the Development of Abstract Reasoning and Performance in an Introductory Programming Class, *Proceedings of the 11th SIGCSE Technical Symposium on Computer Science Education*, Kansas City, USA, 1980, 110-117.

[14] Leeper, R. R., & Silver, J. L. Predicting Success in a First Programming Course, *Proceedings of the 13th SIGCSE Technical Symposium on Computer Science Education*, Indianapolis, USA, 1982, 147 – 150.

[15] Nguyen, D. & Wong, S. OOP in Introductory CS: Better Students Through Abstraction, *Proceedings of the fifth Workshop on Pedagogies and Tools for Assimilating Object-Oriented Concepts*, OOPSLA 2001.

[16] Or-Bach, R. and Lavy, I. Cognitive Activities of Abstraction in Object Orientation: An Empirical Study. *SIGCSE Bulletin*, 36 (2), 2004, 82-86.

[17] Rountree, N. Rountree, J. and Robins, A. Predictors of Success and Failure in a CS1 Course. SIGCSE Bulletin, vol. 34 (4), 2002, 121-124.

[18] Shayer, M. and Adey, P. Towards a Science of Science Teaching, *Heinemann Educational Publishers*, Oxford, England, 1981.

[19] Sprague, P., & Schahczenski, C. Abstraction the Key to CS1. *J.Comput.Small Coll., 17* (3), 2002, 211-218.

[20] Ting, S. R., & Robinson, T. L. First-Year Academic Success: A Prediction Combining Cognitive and Psychosocial Variables for Caucasian and African American Students, *Journal of College Student Development*, 39, 1998, 599-610.

[21] Ventura, P. R. *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.

[22] Ventura, P. R. & Ramamurthy, B. Wanted: CS1 Students. No Experience Required, *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, Norfolk, USA, 2004, 240-244.

[23] Ventura, P.R. Identifying Predictors of Success for an Objects-First CS1, *Journal of Computer Science Education*, 15 (3), 2005, 223-243.

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