Mental Models and Programming Aptitude

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

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

Kasper Dalgaard Larsen
Department of Computer Science
University of Aarhus
Aabogade 34
DK-8200 Aarhus N, Denmark
larsen@daimi.au.dk

ABSTRACT

Predicting the success of students participating in introductory programming courses has been an active research area for more than 25 years. Until recently, no variables or tests have had any significant predictive power. However, Dehnadi and Bornat claim to have found a simple test for programming aptitude to cleanly separate programming sheep from non-programming goats. We briefly present their theory and test instrument.

We have repeated their test in our local context in order to verify and perhaps generalise their findings, but we could not show that the test predicts students' success in our introductory programming course.

Based on this failure of the test instrument, we discuss various explanations for our differing results and suggest a research method from which it may be possible to generalise local results in this area. Furthermore, we discuss and criticize Dehnadi and Bornat's programming aptitude test and devise alternative test instruments.

Categories and Subject Descriptors

K3.2 [**Computers & Education**]: Computer and Information Science Education – *computer science education, information systems education.*

General Terms

Experimentation, Human Factors.

Keywords

Objects-first, CS1, introductory programming, object-oriented programming, predictors of success.

1. INTRODUCTION

In a teaser email circulated in late 2005, shortly before the PPIG workshop in January 2006, Richard Bornat wrote: "We have a scientific breakthrough that we'd like to announce at your little PPIG. The breakthrough is that Saeed has a test which picks out, with 100% accuracy, those people who have a chance of learning to program and rejects, with 100% accuracy, those who have no chance. Don't believe it? Neither did I, at first, but it's true. And I'm not telling you, before the little PPIG, just how it's done. But of course I will tell you all there."

ITiCSE 2007, June 25–27, 2007, Dundee, Scotland.

We learned about the test in conjunction with the PPIG workshop in January 2006. Having searched for predictors of success for introductory programming courses, we were certainly intrigued by the promotion material, and we decided to try to verify Dehnadi and Bornat's findings.

This paper describes what we found. As has already been noted in the abstract, we have not been able to verify Dehnadi and Bornat's findings.

In the next section, we provide a brief overview of some of the research which aims at finding predictors of success for computer science studies at universities. In section 3, we describe the programming aptitude test developed at Middlesex University by Dehnadi and Bornat and their preliminary results. In sections 4 and 5, we present our research method and our findings, which we discuss in section 6. Section 7 provides a succinct conclusion.

2. RELATED WORK

There has been a substantial amount of research conducted to identify general variables that are predictors of the success of students aiming for a degree in computer science. Investigated variables encompass gender [21, 22], ACT/SAT scores [9], students' mathematical abilities [6, 19, 22], performance in prior courses [12], emotional factors [11], abstraction ability [3], and students' own beliefs [24]. Research has also been conducted in the more specific area of introductory programming [2, 5, 8, 10, 17, 18, 20].

Evans and Simkin [15] sum up the arguments given in many studies for performing this kind of study:

1. Discriminating among enrolment applicants
2. Advising students on majors
3. Identifying productive programmers
4. Identifying employees who might best profit from additional training
5. Improving computer classes for non-CIS majors
6. Determining the importance of oft-cited predictors of computer competency such as gender or math ability
7. Exploring the relationship between programming abilities and other cognitive reasoning processes

Dehnadi and Bornat [1, 13, 14] claim they have found a way to identify students who will not succeed in learning programming. Based on a test of 60 students, they claim "[w]e have found a test for programming aptitude, of which we give details. Remarkably, we can predict success or failure even before students have had

any contact with any programming language, and with total accuracy." [14].

Our rationale for conducting this study is primarily to verify the claims made by Dehnadi and Bornat and to build up knowledge about factors that influence students' learning of programming.

3. TEST FOR PROGRAMMING APTITUDE

In this section, we describe the programming aptitude test developed at Middlesex University by Dehnadi and Bornat and their preliminary results as reported in [14].

Dehnadi and Bornat classified students according to their consistency in answering a set of similar questions. The overall hypothesis is that consistent students and consistent students only will be able to learn to program.

To determine consistency, Dehnadi and Bornat used a questionnaire with 12 small Java programs. Each program consists of two variable declarations and one, two, or three assignment statements; Figure 1 shows a sample.

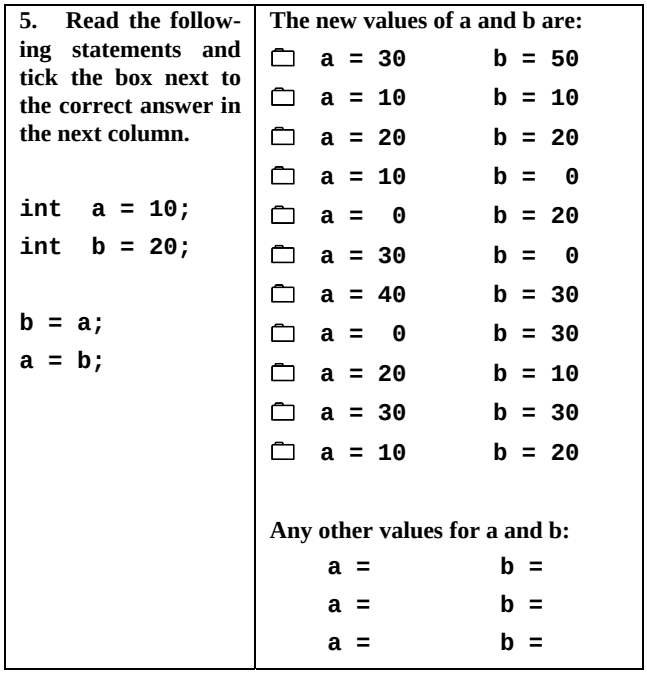

Figure 1: *A sample question from Dehnadi and Bornat's questionnaire*

Dehnadi and Bornat have identified 11 different mental models which are captured by options in the questionnaire (along with the last option: *other*). The questionnaire contains 12 questions similar to the one in Figure 1, giving rise to a 12-tuple describing the mental models applied by a student (e.g. (m_7, m_3, ..., m_7)) where m_i represents a mental model. The 12-tuple is used to *assign* each student to one of three categories:

- The *consistent* group. The students who use the same mental model for most of the questions (irregardless of which model).
- The *inconsistent* group. The students who use varying mental models for the questions.
- The *blank* group. The students who refuse to answer the questions.

In [13], the authors write: "The consistent/inconsistent/blank assignment which is the basis of our preliminary result was rather subjective". In [13], the authors develop a more objective instrument for categorisation of the students—an instrument which we shall use in our investigation.

Dehnadi and Bornat found that 44% of their students belong to the consistent group, and 39% belong to the inconsistent group; 8% left the questionnaire blank (the remaining 9% are missing).

In [14], the authors conclude that the test, although not perfect, is the first test to be able to claim any degree of success:

> "[Our analysis] shows that the first administration of Dehnadi's test reliably separated the consistent group, who almost all scored 50 or above, from the rest, who almost all scored below 50, with only 4 out of 27 false positives in the consistent group and 9 out of 34 false negatives in the rest [...]. Clearly, Dehnadi's test is not a perfect divider of programming sheep from non-programming goats. Nevertheless, if it [was] used as an admissions barrier, and only those who scored consistently were admitted, the pass/fail statistics would be transformed. In the total population 32 out of 61 (52%) failed; in the first-test consistent group only 6 out of 27 (22%). We believe that we can claim that we have a predictive test which can be taken prior to the course to determine, with a very high degree of accuracy, which students will be successful. This is, so far as we are aware, the first test to be able to claim any degree of predictive success."

It is indeed very interesting *if* Dehnadi and Bornat have found a predictive test as they describe.

4. RESEARCH METHOD

In this section, we discuss the methodology used in our study.

4.1 Hypothesis

In this study, we examined the predictive power of a student's mental model for his or her success in learning introductory programming; the hypothesis is that there is a positive correlation between a student's mental model and the student's ability to learn programming. The specific research question we investigated is the following:

> *Is there a correlation between the students' consistency in the mental model applied in questionnaire and their performance in the final exam of a seven-week introductory, model-based, object-oriented programming course?*

4.2 The Course

The programming course spans the first half of CS1 at the University of Aarhus. The course runs for seven weeks; two weeks after the course ends, there is a lab examination with binary pass/fail grading. The grading is based solely upon the final examination; acceptable performance during the course is a prerequisite for the final exam but does not count as part of the grading.

Aims: The purpose of the course is for students learn the foundation for systematic construction of simple programs and, through this, obtain knowledge about the role of conceptual modeling in object-oriented programming. The goal is that students become familiar with a modern programming language, fundamental programming language concepts, and selected class libraries.

Competencies: After the course, students should be able to explain and use fundamental elements in a modern programming language, use conceptual modelling in relation to preparing simple object-oriented programs, implement simple object-oriented models in a programming language, and use selected class libraries.

Form: The course runs for seven weeks; every week, there are four lecture hours and four lab hours with a TA. In addition to the scheduled hours, students are supposed to work approximately seven hours per week in study groups or on their own. There is a weekly mandatory assignment.

Exam: The examination resembles an ordinary lab session. The students are tested in groups of up to 25 at a time. The effective examination time is 30 minutes (occasionally, for various reasons, we allowed a bit more time); a full hour is scheduled for each group to allow for preparing and finalizing (upload, etc.). Each group receives a different assignment consisting of 10 small progressive programming tasks. In principle, the assignments are identical (they are all instances of the same generic assignment).

There are *two checkpoints* in the assignment: one after task three and one after task eight. The students are instructed to call upon an examiner to demonstrate their solutions when they reach either of the checkpoints. For each student, we noted the elapsed time at both checkpoints as well as when (if) they finished the assignment (*first interval*, *second interval*, and *final time*), thus providing a rough measure of the student's efficiency and competence.

A more detailed description of the course can be found in [7]; an evaluation of the examination can be found in [4].

4.3 Subjects

There are approximately 300 students from a variety of study programmes, e.g. computer science, mathematics, geology, nano science, economy, multimedia, etc. Forty percent of the students are majors in computer science; they are the only group of students that continues 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 population for this study was 142 students; of the 150 students who volunteered to participate at the beginning of the course, 142 attended the final exam.

The students answered the questionnaire in the first week before the assignment statement was taught.

4.4 Classification of Mental Model and Exam Result

To determine the consistency of the mental model for each of the students, we used the categorization instrument proposed by Dehnadi [13]. From the 12-tuple that describes the mental models applied by a student in the questionnaire, we divided the students into five categories C_i, $0 \le i < 5$, of decreasing consistency, C_0 being the most consistent category and C_4 the least consistent category. A student is in consistency category C_0 if at least eight mental models in the student's 12-tuple are identical. For the coarse-grained consistent/inconsistent categorization, students in C_0 are considered consistent while students in any of the other categories are considered inconsistent. For further details, see [13].

The binary pass/fail grading of the exam was too coarse-grained to allow for statistical analysis. Therefore, we subdivided the students into four groups, G_i, $0 \le i < 4$. G_0 represents the students that failed the exam; G_1 represents the students who barely passed the exam (i.e. reached the second checkpoint in the very last minute), G_2 represents the students who produced an average performance (i.e. reached the second checkpoint in due time but did not finish the assignment), and G_3 represents the students who finished the assignment within the time limit with a program that fulfils the complete specification.

5. FINDINGS

In this section, we present the findings from the questionnaire.

5.1 Results

The distribution between consistent and inconsistent broken down to the exam result and prior programming experience is shown in Table 1.

	Consistent	**Inconsistent**
Total	124	18
Pass at the final exam	120	16
Fail at the final exam	4	2
Prior programming experience	85	2
No prior programming experience	39	16

Table 1: *Number of consistent and inconsistent students*

We might consider breaking the data down to other variables, e.g. gender, major, and seniority (study age); however, from previous research, we know that these do not influence students' performance in this course [6]).

5.2 Programming Aptitude

In order to validate Dehnadi and Bornat's findings, we have used a Pearson correlation coefficient test [23] to find if, for students with no prior programming experience, there is a significant correlation between the consistency level and the grading level (according to the *C*- and *G*-categories described in section 4.4).

The P-value is –0.072. Thus, we concluded that there is *no correlation* between consistency of the mental model and performance in our introductory programming course, i.e. we cannot verify Dehnadi and Bornat's findings. Traditionally, a P-value of at least 0.3 (numerically) is required for correlation. (The negative P-value is expected since C_0 corresponds to the highest level of consistency and C_4 to the lowest level of consistency.)

To take a closer look at this contradictory result, we have tested for correlation for a more fine-grained partitioning than the five competence-levels and four grading levels applied above.

We made a more fine-grained partitioning of the mental models by refining the C_i categories: C_i represents the students' whose maximum number of answers of the same mental model equals i, thus providing 13 different categories of mental models. Similarly, we have refined the G_i categories to reflect the students' performance according to the second interval (the time elapsed when reaching the second checkpoint), i.e. G_i is the students for whom the second interval is i minutes.

The distribution of the data certainly does not indicate a correlation (see Figure 2). A Pearson correlation test confirms this impression with the same result as before (P=–0.075).

Our result is a clear and unequivocal rejection of the research question: there is absolutely no correlation between students' consistency of the mental model applied in the questionnaire and

their performance in the final exam of a seven-week introductory, model-based, object-oriented programming course.

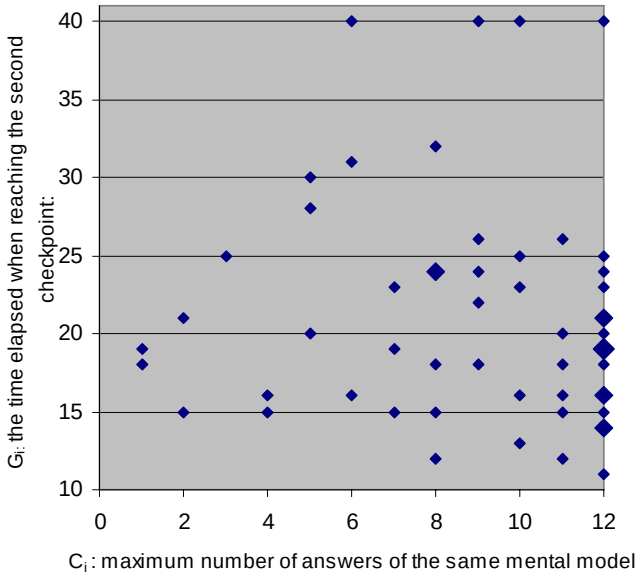

Figure 2: *Second interval versus maximum number of identical mental models in 12-tuple*

If the hypothesis of positive correlation between a student's mental model and ability to learn programming is to be confirmed, it requires an interpretation of the mental model which is different than the one reflected in Dehnadi's questionnaire or another interpretation of ability to learn programming different than the one reflected by the exam of the introductory programming course.

6. DISCUSSION

Our unequivocal result gives rise to a number of questions. One question is whether Dehnadi and Bornat's interpretation of their results is viable. Another question is the validity of the test instrument and speculations about other and better test instruments. But before we address these questions, let us look at possible explanations of our differing results.

6.1 Explaining the Differing Results

The large number of non-fixed variables in Dehnadi and Bornat's investigation and ours allow many explanations. First of all, the investigations have been carried out in different course contexts. We do not know much about the nature of the course at Middlesex University, but it is undoubtly different than ours and may be so in many respects: course material (e.g. textbook, programming language, development environment), course structure (e.g. number of lectures and lab hours), course work (e.g. mandatory assignments, project work), availability of resources (e.g. support material, support for collaboration, student/instructor ratio), and the degree of alignment (concordance between syllabus, course content and the exam). Also, the exam may be different; again, we do not know anything about the nature of the courses at Middlesex University. The instructor is different and may be so in many respects (e.g. teaching experience, familiarity with the subject, personal attitude), and, finally, the students may be different in many ways (e.g. age, study seniority, major).

With all this variation, how can we ever generalise findings from the context where the findings are identified? The best way is by inductive reasoning which can be fuelled by similar findings across a multitude of institutions [16].

6.2 Questioning the Validity of the Test Instrument

Dehnadi and Bornat's interpretation of the students' behaviour in the first test goes as follows: "What distinguish the three groups in the first test is their different attitudes to meaninglessness. The consistent group showed a pre-acceptance of this fact: they are capable of seeing mathematical calculation problems in terms of rules, and can follow those rules wheresoever they may lead. The inconsistent group, on the other hand, looks for meaning where it is not. The blank group knows that it is looking at meaninglessness, and refuses to deal with it."

Contrary to Dehnadi and Bornat, we interviewed our subjects. We conducted individual interviews with the 14 students who were inconsistent but did pass the final exam. They all remembered the test very well. Interestingly, they all started out with some mental model, some set of rules that gave meaning to the "meaningless" notation in the questionnaire. The problem for the 14 students was that the model they started out with failed at some point before the end of the test. Not knowing about the purpose of the test, and not considering it important, none of the students cared to backtrack to find a viable model. They simply altered their model and went on from there. Our harsh conclusion is that it seems as if the only thing the test instrument is testing is the students' guessing capabilities; can they guess a *viable* model up front or can they not? This is hardly an interesting classification of students.

6.3 Alternative Test Instruments

In the light of the conclusion of the previous section, we must reject the test instrument proposed by Dehnadi and Bornat. However, the idea of testing a correlation between the inclination to give meaning to meaninglessness and performance in an introductory programming course, as suggested by Dehnadi and Bornat's comment in their interpretation of their observations, hints at an alternative test instrument.

If the hypothesis is that the inclination to give meaning to meaninglessness is a predictor of success in an introductory programming course, we should devise a test instrument for that. Such a test instrument can easily be constructed by describing a meaningless set of rules and then asking the students to apply these rules to a number of situations. Different test instruments could be constructed: some that invite for interpretation and resulting false applications of the rule set, and some that (by being more neutral) does not. Developing different test instruments along these lines enables tests of the test instruments which in itself is a reasonable task to undertake.

7. CONCLUSION

We tested the hypothesis of a correlation between a student's mental model (according to Dehnadi's definition in [14]) and how well the student performs in an introductory programming course at university. Our result is an unequivocal rejection of the hypothesis.

The result is a surprise—at least in light of [14] in which the authors conclude that the test, although not perfect, is the first test to claim any degree of success.

We have enumerated many explanations for our differing results; in particular, we question the test instrument from [13] used to categorise students according to the mental model, and we suggest a research method from which it may be possible to generalize local results in this area.

A qualitative analysis in the form of interviews with selected subjects has revealed that the test instrument does not seem to measure what it is supposed to; based on insights from the interviews we have devised alternative test instruments.

Our result is encouraging since we do not adhere to the sheep-goat presumption about programming aptitude. To the extent that we shall ever be able to identify concrete factors that predict success, we will use these to improve students' background to increase their chances for success in learning to program. That is our motivation for doing research in this area.

Dehnadi and Bornat's idea of predicting success from mental model is interesting and maybe viable but at least requires an improved test instrument.

8. ACKNOWLEDGMENTS

We thank all the students from the course *Introduction to Programming* at the University of Aarhus in the fall of 2006 who took the time to participate and make this research possible.

9. REFERENCES

[1] C. Arthur. How can I tell if I'll be any good as a programmer? In *The Guardian*, Thursday July 27, 2006.

[2] J. Bennedsen. Teaching Java programming to media students with a liberal arts background. In *Proceedings for the 7th Java & the Internet in the Computing Curriculum Conference* (JICC 7) Monday 27th January 2003, 2003.

[3] J. Bennedsen & M. Caspersen. Abstraction ability as an indicator of success for learning object-oriented programming? *SIGCSE Bulletin*, 38(2):39-43, 2006.

[4] J. Bennedsen and M. Caspersen. Assessing Process and Product - A Practical Lab Exam for an Introductory Programming Course. In *Proceedings of the 36th Annual Frontiers in Education Conference*, M4E-16-M4E-21, San Diego, California October 28-31, 2006.

[5] J. Bennedsen and M. Caspersen. An Upcoming Study of Potential Success Factors for an Introductory Model-Driven Programming Course. In *Proceedings for the Fifth Koli Calling Conference on Computer Science Education* pages 166-169, Koli, Finland 18-20 November 2005.

[6] J. Bennedsen and M. E. Caspersen. An investigation of potential success factors for an introductory model-driven programming course. In *ICER '05: Proceedings of the 2005 International Workshop on Computing Education Research*, 155-163, Seattle, WA, USA, 2005.

[7] J. Bennedsen and M. E. Caspersen. Programming in context: a model-first approach to CS1. In *SIGCSE '04: Proceedings of the 35th Technical Symposium on Computer Science Education*, 477-481, Norfolk, Virginia, USA, 2004.

[8] S. Bergin and R. Reilly. Programming: factors that influence success. In *SIGCSE '05: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 411-415, St. Louis, Missouri, USA, 2005.

[9] D. F. Butcher & W. A. Muth. Predicting performance in an introductory computer science course. *Communications of the ACM*, 28(3):263-268, 1985.

[10] P. Byrne and G. Lyons. The effect of student attributes on success in programming. In *ITiCSE '01: Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, 49-52, Canterbury, United Kingdom, 2001.

[11] C. G. Cegielski & D. J. Hall. What makes a good programmer? *Communications of the ACM*, 49(10):73-75, 2006.

[12] A. T. Chamillard. Using student performance predictions in a computer science curriculum. In *ITICSE '06: Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education*, 260-264, Bologna, Italy, 2006.

[13] S. Dehnadi. Testing programming Aptitude. In *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group*, 22-37, Brighton, UK, 2006.

[14] S. Dehnadi and R. Bornat. The camel has two humps. 2006. www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

[15] G. E. Evans & M. G. Simkin. What best predicts computer proficiency? *Communication of the ACM*, 32(11):1322-1327, 1989.

[16] S. Fincher and M. Petre. *Computer Science Education Research*. Routledge Falmer, London, 2004.

[17] D. Hagan and S. Markham. Does it help to have some programming experience before beginning a computing degree program? In *ITiCSE '00: Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education*, 25-28, Helsinki, Finland, 2000.

[18] R. R. Leeper and J. L. Silver. Predicting success in a first programming course. In *SIGCSE '82: Proceedings of the Thirteenth Technical Symposium on Computer Science Education*, 147-150, Indianapolis, Indiana, United States, 1982.

[19] L. P. McCoy & J. K. Burton. The relationship of computer programming and mathematics in secondary students. *Comput. Sch.* 4(3-4):159-166, 1988.

[20] N. Pillay & V. R. Jugoo. An investigation into student characteristics affecting novice programming performance. *SIGCSE Bulletin*, 37(4):107-110, 2005.

[21] N. Rountree, J. Rountree, A. Robins and R. Hannah. Interacting factors that predict success and failure in a CS1 course. In *ITiCSE-WGR '04: Working Group Reports from Innovation and Technology in Computer Science Education*, 101-104, Leeds, United Kingdom, 2004.

[22] P. Ventura. Identifying predictors of success for an objects-first CS1. *Computer Science Education*, 15(3):223-243, 2005.

[23] L. Wallnau and F. Gravetter. *Essentials of Statistics for the Behavioral Sciences*. Thomson Learning, New York, 2005.

[24] B. C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. In *SIGCSE '01: Proceedings of the thirty-second technical symposium on Computer Science Education*, 184-188, Charlotte, North Carolina, United States, 2001.