An Investigation of Potential Success Factors for an Introductory Model-Driven Programming Course

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

In order to improve the course design of a CS1 model-driven programming course we study potential indicators of success for such a course. We explain our specific interpretation of objects-first.

Of eight potential indicators of success, we have found only two to be significant at a 95% confidence interval: math grade from high school and course work. The two significant indicators explain 24.2% of the variation of the exam grade. The result concerning math grade contradicts earlier findings.

We discuss four aspects of our research: the explanation power of the potential success indicators, the impact of our findings on teaching, limits of what to conclude from the available data, and the variety of the notion "objects-first".

Because of the variety of interpretations of "objects-first", the present research is necessary as a supplement to earlier research in order to make generalizable results on the success factors for objects-first programming.

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, object-oriented programming, model-driven programming, predictors of success, course design.

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 gender [23], the educational level of parents [26], ACT/SAT scores [5, 12, 23], and emotional factors [25]. Research has been conducted in the general context of education, within computer science, and in the more topic specific area of introductory programming [4, 6, 13, 17, 19]. 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 Venture [27] focuses on a systematic evaluation of hypothesis related to the factors for success of an introductory programming course using an objects-first approach [14]. The results are documented in [27, 28].

Ventura analysed different factors and their influence on the outcome of participation in an object-first CS1 course. The predictors included prior programming experience, mathematical ability, academic and psychological variables, gender, and measures of student effort [27 p. xxi]. Ventura's conclusion is that there are big differences between the previous findings in imperative-first programming courses and his object-first programming course. In the studies of imperative-first courses the student's mathematical abilities was found to be a predictor of success [17]. Ventura [27], however, found that this is not the case in his object-first CS1 course. In the imperative-first course, prior programming experience was also a predictor for success; Ventura found this was not the case in his objects-first course. As a curiosity he found that previous knowledge of Java was a negative predictor of success: the students with previous knowledge performed worse than students without [27 p. 73].

As always there are some preconditions to the research. One important precondition is the characteristics of the course that founded the basis for the research. Ventura used a CS1 course with a graphics early approach. In [28] he describes the graphics early approach as follows: The course focuses primarily on the teaching of problem solving using object-oriented design techniques with the following features [28 p. 241]:

Design-centered. Through the introduction of a simplified version of UML class diagrams, students are taught to think about problem solutions independently of the code. Design once, code anywhere has become the motto for the class. Design patterns are introduced both in lecture and integrated into the programming assignments. These serve as examples of good design as well as vehicles to encourage students to think at a higher level of abstraction.

Graphical. Classroom examples use graphics to motivate and ground OO concepts such as encapsulation, inheritance, and polymorphism. The programming assignments are also graphical allowing the students to build programs that are like those they are used to using.

Objects-first. Students are taught from the very beginning to think in terms of objects and the fundamentals of object-oriented programming, encapsulation, inheritance, and polymorphism. These

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICER'05, October 1–2, 2005, Seattle, Washington, USA.
Copyright 2005 ACM 1-59593-043-4/05/0010...$5.00.

concepts are introduced before traditional language constructs for selection and iteration.

Furthermore, Ventura writes: "Empirical testing was conducted on the graphical design-centric objects-first CS1 to identify the predictors of success" [28 p. 127]. Venturas findings are only valid for students participating in an introductory programming course similar to his. In the current research, we look for potential success factors for an introductory programming using a different approach than Ventura's; our approach is best characterized as a model-based approach to programming [1].

2. A MODEL-DRIVEN PROGRAMMING COURSE

This section describes goal, form, and content of the model-driven programming course as well as the lab test that constitutes the final examination and upon which the grading is based.

2.1 General Information

This course constitutes the first half of CS1 at University of Aarhus. The course runs for seven weeks, one to two weeks after the course there is a lab test with a binary pass/fail grading.

The grading is based solely upon the behaviour in and result of a lab test; suitable performance during the course is a prerequisite for the final exam but does not count as part of the grading.

There are approximately 235 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 students are grouped in teams of 18-20 students; in the fall of 2004 there where 13 teams. Each team has its own Teaching Assistant (TA).

2.2 Goals

The purpose of the course 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.

After the course the student will 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 OO-models in a modern programming language, and use selected class libraries.

2.3 Form

The course runs for seven weeks; every week there are four lecture hours[1], one lab hour with a TA, and three class hours also with a TA. Besides scheduled hours, the students are supposed to work approximately seven hours per week in study groups or on their own.

[1] For scheduled actvities (lectures, labs, classes, etc.) an hour means only 45 minutes.

Every week (except for the first), there is a mandatory assignment that must be handed in to the TA. The TA examines the assignments and gives personal as well as collective feedback to the students. If an assignment is too weak, the student gets a chance to improve it. Approval of five out of six weekly assignments is a prerequisite for the final exam but does not count as part of the grading.

The four lecture hours per week are used for presentation and discussion of general concepts and specific details in the course material, but also for live programming. Live programming is programming in front of the students in the lecture theatre using computer and projector. The purpose of live programming is to reveal the programming process to the students (see [2]).

The one lab hour per week is unstructured in the sense that the students (typically in pairs) work on what they find useful, the purpose of the lab is that the students can get help from a TA while working on the exercises of the week.

The three class hours per week are used for discussion of the weekly assignment, for discussion of other exercises that the students has been working on, as well as for discussion of topics from the textbook.

For the coming versions of the course, we are planning to adopt closed labs as a more structured form of the lab activities; also we are planning to reschedule such that two (or three) hours per week are spent on closed labs and two (or one) in the classroom.

2.4 Contents

The course content is fundamental programming language concepts, object-orientation, and techniques for systematic construction of simple programs.

- *Fundamental programming language concepts*. Variable, value, type, expression, object, class, encapsulation, control structure, method/procedure, recursion, type hierarchies.
- *Object-orientation*. Modelling; class structures (specialization, aggregation and association); use of selected class libraries (in particular collection libraries), interfaces and abstract classes.
- *Systematic development of small programs*. Modularization, stepwise refinement/incremental development, test.

The above is a logical listing of the course contents; it is *not* the order in which the content is covered. The content is covered using a spiral approach [3]; for further details on the structure and contents of the course, see [1, 7, 8].

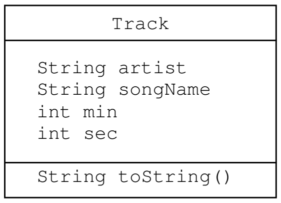
2.5 The Exam

The exam is organized such that 20 students are tested concurrently. The test takes place in a lab; besides the 20 students, five TAs, the lecturer and an external examiner are present in the lab.

We schedule one hour per group of 20 students, but only 30 minutes are used for the lab test. The rest of the time is used for administrative activities and as a buffer.

Lab Test Exercise (30 minute test)

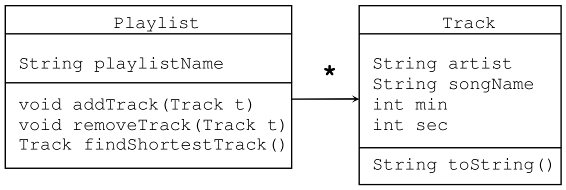
1. Create a class, *Track*, that represents a piece of music; the *Track* class is specified in the following UML diagram.



The four field variables must be initialized in a constructor (through four parameters of suitable types). The method *toString* must return a string representation for a piece of music, e.g.

```
"Yesterday: The Beatles (2:05)"
```

2. Create a test method named *exam* in class *Driver*. The method must be static, have return type void, and have no parameters.
3. Create two *Track* objects in the exam method using object references *t1* and *t2*; print the two *Track* objects using the *toString* method.
4. Create a new class, *Playlist*, representing a collection of *Track*s; the *Playlist* class and its relation to the *Track* class is specified in the following UML diagram:



5. Implement the method *addTrack* (and *removeTrack*) so that it adds (removes) the object *t* to (from) the *Playlist* object.
6. Create a *Playlist* object in the *exam* method in the *Driver* class; associate the two existing *Track* objects with the *Playlist* object.
7. Implement the method *findShortestTrack*. The method must return a shortest (measured in playing time) *Track* object from a *Playlist* object. You can assume a non-empty *Playlist* object. In other words: You need not worry about the *playlist* being empty.
8. Use the methods *findShortestTrack* (from class *Playlist*) and *toString* (from class *Track*) to print the shorter of the two *Track* objects that was created and associated with the *Playlist* object in the *exam* method.
9. Let the *Track* class implement the *Comparable* interface. The natural order of *Track* objects is defined as alphabetical order of *artist* (secondary of *songName*).

Figure 1: *Sample Lab Test Exercise*

Each group of students get a different assignment. In principle the assignments are identical (they are all instances of the same generic assignment), but the students does not know nor realize this. The similarity of the assignments is important for fairness as well as comparability of the students' results. A sample assignment is presented in figure 1.

At the beginning of the exam, the students get a sheet of paper with the assignment consisting of nine small progressive programming tasks, and then they start programming. To ease inspection, we tell the students to tile all editor windows on the screen during the test.

When the first three tasks are finished, the students must demonstrate what they have achieved for one of the TAs. The lecturer and the external examiner evaluate the process as well as the product of each student, i.e. the students behaviour as well as the quality of the programs they produce counts in the final grading.

The sample lab test exercise in figure 1 is about tracks and playlists; the other exercises are about luggage and flights, employees and departments, side effects and medicine, etc. Although the concepts modelled by the classes vary, the assignments have similar structure. Because of this similarity, it is very easy for the lecturer and the external examiner without too much effort to evaluate the achievements of each student.

2.6 Apprenticeship Inspired Pedagogy

The course utilizes an apprenticeship-based pedagogy where students are exposed to how an expert programmer works. This is implemented by the lecturer behaving as a professional programmer (the master). For more information see [11].

The master reflects and thinks aloud of the particular action, maing them visible and as a source of identification [19]. As such, the apprentice (student) learns from observing the master (teacher) performing the actions embedded in the profession (e.g. coding, testing, etc).

3. RESEARCH METHOD

This paragraph discusses the methodology utilized in identifying the predictors of success for the model based CS 1 course described in the previous section. Section 3.1 outlines the research questions to be studied. Section 3.2 provides details on the subjects involved in the study. Section 3.3 describes the data and how it was provided, while Section 3.4 presents the manner in which data were collected and calculated.

3.1 Research Questions

In our current research, we look for potential success indicators that are statistically significant in predicting students' success when undertaking a model-driven introductory programming course. The factors are motivated by previous research in the field [13, 17, 27, 30].

1. What is the relationship of mathematical ability to model-based CS1?
2. What is the relationship of gender to model-based CS1?
3. What is the relationship of major/intended major to model-based CS1?
4. What is the relationship of course work to model-based CS1?
5. What is the relationship of years at the university to model-based CS1?
6. What is the relationship of the team to model-based CS1?

Due to technical problems, we did not collect data on the students feeling about the course, motivation for the course etc. Due to a technical problem, we were not able to use information about the students' previous programming experience.

3.2 Subjects

The subjects studied in this paper were students enrolled at the course Introduction to Programming at the University of Aarhus, Denmark, during the fall of 2004. Only data from students taking the course for the first time were used; to exclude the possibility of an extended practice effect, we decided to exclude from our investigations the students who followed the course for the second or third time.

3.3 Data

Several different data sources were used in this study. Information comes from the administrative system at the university (gender, enrollment date, major), the course web-site (team number), the teaching assistants (the score of the different lab-assignments), the final exam and the authors (the score in the exam) and a questionnaire (the math score for their high school exam).

Mathematical ability. The students score from their high school exam is used as an indicator of the students' mathematical abilities. The high schools in Denmark offer different levels of mathematical exams (A, B, and C where C is the lowest level). The students are required to have a high school math exam at the A level in order to take the introductory programming course. However, three students did not have the required A level but a B level. In our analysis, we observe that these three persons are outliers very far from the normal distribution, so they are excluded from the analysis. The students themselves in a questionnaire gave the score after the exam. A few students did not answer the questionnaire; they are also excluded from the analysis.

Course work. During the course, the students are required to complete five out of six weekly exercises in order to participate in the final exam. The teaching assistants evaluate the exercises and the score for each exercise is encoded as one of the numbers 1, 2, or 4. The interpretation of the encoding is:

Value	*Meaning*
1	Perfect, no significant errors
2	OK, small errors
4	Not accepted/Not handed in

Table 1: The scores for the weekly exercises

In case a student got a "4", he had the possibility of resubmitting the exercise once.

We have used the sum of these scores as a description of the students work during the course. We have excluded from our analysis the students who were not allowed to take the final exam.

Final exam score. The final exam is a practical test as described in section 2. 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, one author has post-marked all the students' solutions. The result of the more fine-grained marking is a grade in the interval [00...13] (see [10]). In order to pass an exam, a student needs a grade of 6 or more.

The official description of the grades is [10]:

13: Is given for the exceptionally independent and excellent performance.

11: Is given for the independent and excellent performance.

10: Is given for the excellent but not particularly independent performance.

9: Is given for the good performance, a little above average.

8: Is given for the average performance.

7: Is given for the mediocre performance, slightly below average.

6: Is given for the just acceptable performance.

5: Is given for the hesitant and not satisfactory performance.

03: Is given for the very hesitant, very insufficient and unsatisfactory performance.

00: Is given for the completely unacceptable performance.

The results of the post-marking is equivalent to the official results of the exam in the sense that all the students who passed the exam got a grade of six or more and the students who failed the exam got a grade of five or less. In order to ensure that the marking was fair, the co-author marked ten randomly selected answers. The results were identical.

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

3.4 Statistical Analysis

In order to test the hypotheses a covariance analysis is used. The analysis shows which (if any) of the independent variables that are correlated with the exam result.

The goal is furthermore to find how much impact (if any) the variables have on the result of the examination. One way to obtain this is to use a multiple regression analysis based on an as simple as possible model using the variables in question and the relevant interaction variables (i.e. combination of the variables).

In order to test the multiple regression model normally six prerequisites need to be fulfilled:

1. Linearity
2. Normal distribution
3. Homoscedasticity – the conditional distribution of Y has constant standard deviation throughout the range of values of the explanatory variables.
4. No collinearity – two or more variables have a strong linear relationship (i.e. explains the same).
5. No problematic outliers – an observation falls far from the rest of the data and the mean is highly influenced.
6. No autocorrelation - some observations are dependent.

Being population data, the requirements are not as important as if they were test samples. The team and the intended major have a correlation. This is to be expected since the teams are made up mostly of students with the same intended major. Team is therefore excluded from the analysis. The data fail on the test for normal distribution, but the test for 3, 4, and 5 is fine. Test 6 is only relevant for time series data. It is therefore possible to use multiple regression analysis in order both to check the hypotheses and furthermore to evaluate the impact the selected factors have on the actual exam result.

We start by running the complete multiple regression model with all variables including all the interaction variables. We find that the model explains 36.1% of the variation in the dependent variable at a 95% confidence interval. In order to meet the criteria of parsimony we compare the complete model with all interaction variables with a simple model i.e. a model without interaction variables. We find that we lose 34.4% explanation power since the simple model only explains 23.6% of the variation in the dependent variable. Because of the severe loss of explanation power in the simple model, we cannot ignore the model with all interaction variables. We want all variables in the model to be significant; this leads us to eliminate one by one all insignificant variables at a 95% confidence interval according to the hierarchical principle; we end up with a reduced model that explains 24.2% of the variance.

In the following, a 95% confidence interval is used to test the hypotheses (i.e. the probability of the hypotheses being true is 95%).

The analysis of the data is performed in SPSS version 13.0. The following variables are used:

Name	Description
GRADE	The result of the programming exam. Integer value from 00 – 13.
MATH	The score from the high school math exam. Integer value from 00 – 13.
COURSEWORK	The results of the assignments during the course. Integer value from 0-8. The variable is translated in the following way:

Sum of the results of the weekly assignments	*Value*
6	8
7	7
8	6
9	5
10	4
11	3
12	2
13	1
14	0

	For an explanation of the results of the weekly assignments, see Table 1. If the sum of the assignments is 6, the student has handed in six perfect answers. If the value is 14 the student has handed in five acceptable assignments and one not acceptable/not handed in.
STUDYAGE	The number of years the student has been enrolled at the university. Integer value from 0 – 20. Students enrolled in 1984 or earlier were coded as 20.
COMPSCIENCE	The student intends to major in computer science (1=intended major in computer science, 0 otherwise).
GEOLOGY	The student intends to major in geology (1=intended major in geology, 0 otherwise).
MATHEMATICS	The student intends to major in math (1=intended major in math, 0 otherwise).
NANOSCIENCE	The student intends to major in nanoscience (1=intended major in nonoscience, 0 otherwise).
SEX	1= female, 0=male.

Table 2: *Description of the variables*

4. RESULTS

In this section the result of the multiple regressions is given.

4.1 Non significant variables

The variables NANOSCIENCE, MATHEMATIS, GEOLOGY, COMPSICENCE, SEX and STUDYAGE were not significant with respect to explaining the exam result using a 95% confidence interval. This was also the case with the interaction variables.

4.2 Multiple regression formula

The result of the regression analysis is presented in Table 3. The derived regression formula is:

```
GRADE =    1.118 +
           0.589*MATH +
           0.341*COURSEWORK
```

Variable	Unstandardised coefficients		Significance
	B	Std. Error	
COURSEWORK	0.341	0.097	0.000
MATH	0.589	0.107	0.001

Table 3: *Coefficients of the regression analysis*

The multiple regression formula (The reduced model with just two variables) explains 24.2 % of the variation of the exam grades. As described above the model with all the interaction variables explains 36.1% of the variation. The loss of explanation power in the reduced formula is 32.69%.

In order to find the importance of the different variables we have calculated the squared partial correlation coefficients (r^2). These describe the impact of one of the variables when the other variables are held fixed; in other words the amount of the variation of the exam grade that one of the variables is responsible for.

	r	r^2
COURSEWORK	0.264	0.069696 = 7 %
MATH	0.393	0.154449 = 15,4 %

Table 4: *Partial correlation coefficients*

In order to get a model that explains more of the variation of the exam grades we have tested the complete model using a 90% confidence interval for the individual variables. The reason is that it is population data. This gives the following formula:

```
GRADE =    -13.58 + 2.575*COURSEWORK +
           1.856*MATH + 3.564*COMPSCIENCE +
           6.668*GEOLOGY -
           0.673*MATH*GEOLOGY+
           0.064*MATH*STUDYAGE-
           0.192*COURSEWORK*MATH -
           0.515*COURSEWORK*COMPSCIENCE -
           0.111*COURSEWORK*STUDYAGE.
```

This formula accounts for 29.9% of the variation of the exam grade. In order to be compatible with the references, we will only discuss the model with a 95% confidence interval for the individual variables.

4.3 The hypotheses

In the following, we will discuss the research questions.

4.3.1 *Mathematical ability*

In the multiple regression formula, we can see that the math score from high school has a positive impact on the exam grade. We therefore accept the hypothesis that there is a positive correlation between the final exam score and the grade from the math exam in high school (95% confidence interval).

The squared partial correlation coefficient in the multiple regression was 15.4% saying that math grade alone accounts for over 15% of the variance of the final grade. This is almost the same that Leeper & Silver [17] found in their analysis; they found that math accounted for 14.3% of the variation.

4.3.2 *Gender*

The variables SEX was not significant, neither at the 95% confidence interval nor at the 90% confidence interval. We can therefore not accept the hypothesis that gender has an impact on the exam score. This corresponds with the findings of Ventura [28] "The tests fail to reveal any gender bias for course success." (p. 98), and the findings of [22].

4.3.3 *Major/intended major*

Neither of the variables indicating the intended major of the students were significant at the 95% confidence interval. This implies that we must reject the hypothesis of a positive impact of majoring in computer science. In the less accurate model, where the variables only were significant at the 90% confidence interval, the variables COMPSCIENCE and GEOLOGY were significant. At this level we can accept the hypothesis of a positive impact of majoring in computer science (it accounts for 3,6% of the variance), but since the variable GEOLOGY is significant but the variables NANOSCIENCE and MATHEMATICS are not, we can not say anything about the students not majoring in computer science. The finding corresponds with the findings in [27].

4.3.4 *Course work*

From the multiple regression formula, we can see that the variable COURSEWORK is significant at the 95% confidence interval and it has a positive impact on the exam grade. We can therefore accept the hypothesis that students who work harder get better grades.

The squared partial correlation coefficient in the linear regression was 7.0% indicating that course work alone accounts for 7% of the variance of the final grade; only half the impact of the math grade from high school.

4.3.5 *Study age*

The variable STUDYAGE was not significant at the 95 % confidence interval. We must therefore reject the hypothesis that there is a correlation between how many years the students have spend at the university and the result of the introductory programming course. Using a 90 % confidence interval, the variable is not significant in itself but in combination with the math grade, it has a positive impact; with course work, it has a negative impact. These two combinations of variables accounts for 2% of the variation each, but this is only at the 90% level.

4.3.6 *Team*

There is an a priori correlation between team and intended major because of the way students are allocated to teams, so the variable team was excluded from the model. Since intended major is not significant, the same is true for team.

5. Discussion

In this section, we discuss four aspects of our investigation: the explanation power of the variables, the impact of our findings on teaching, limits of what to conclude from the available data, and the variety of the notion "objects-first".

5.1 Explanation power of variables

The regression formula presented in section 4.2 accounts for 24.2% of the variation of the exam grade. One way to interpret this is that there is 75.8% not accounted for by these variables, so we cannot predict the actual grade from the two variables. This is the same conclusion that Leeper & Silver [17] reached; they used the regression formula on the students in next year's course and found they were only correct for 39 out of 106 students. On the other hand, we have only used two variables and using these, we can explain 24.2% of the variation of the exam grade – quite a large portion with only two variables. The variables considered here definitely have a large impact on the result of the exam.

5.2 Impact on teaching

The two variables we have found to be significant are math grade from high school and course work. The math grade counts for 2/3 of the explanation power of the two variables but unfortunately the students cannot improve it. Course work counts for 1/3 of the explanation power of the two variables, but opposite to the math grade, course work is improvable in the course and therefore interesting when designing the pedagogy of the course.

The significance of the course work variable indicates, not surprisingly, that students who follow the pace of the course performs better at the final exam. We discuss this aspect further in section 6 on future work.

5.3 Limits of conclusions

Prediction of success is difficult. Ventura [27] reached the conclusion that math score was not a success factor, we have found it to be!

This difference, of course, is a result of the origin of the data. Ventura's [27] and our data come from two different implementations of an objects-first CS1 programming course; we can only draw conclusions for each particular implementation, we cannot draw conclusions about success factors for objects-first programming courses in general. In order to answer the more general question of success factors of an object-first CS1 course we need data from various different implementations of this teaching strategy.

5.4 Objects-first

Objects-first is not a well-defined term. It seems that every CS1 teacher has his or her own interpretation of the term (e.g. 9, 15, 16, 24]. In [14] the description of objects-first is: "an objects-first approach that emphasizes early use of objects and object-oriented design" (p. 28). What does early mean, and what is meant by object-oriented design?

In [18] the author discuss nine myths about object-orientation and its pedagogy; one is that the phrase "objects first" is well.defined. The author writes: "No matter what your definition of objects first is, it is likely to be different from that of the person next to you." (p. 247), and "The phrases 'objects first' and 'objects early' are bandied about in a variety of contexts. When discussing a CSI course they are often used to convey the general idea that objects are discussed early in the course and established as a fundamental concept. Beyond that, however, these phrases seem to take on a variety of meanings, with important implications." (p. 246).

Because of the variety of interpretations of "objects-first", it is impossible to make conclusions about this approach in general.

6. FUTURE WORK

In the current research, we have investigated the relationship between the student's achievements in the final exam of the introductory programming course and mathematical prerequisites, gender, study program, student team, maturity, and the student's achievements in the mandatory weekly exercises during the course.

Identifying success factors is relevant, and has been done in many fields with many different hypotheses of success factors for education in specific fields and for education in general. Wang & Hertel [29] abstracted over more than 11.000 statistical findings in order to identify the most influential factors for learning. They found that "the students metacognitive processes that is, a student's capacity to plan, monitor, and, if necessary, re-plan learning strategies—had the most powerful effect on his or her learning." (p.75). Even though their research was based on students in primary and lower secondary schools, other research have found factors related to student aptitude or classroom management to be important as well in a university setting. For references, see [21].

The explanation power of the variables we have studied is rather small. However, more important is the fact that the most influential of the variables from our study, math grade from high school, is outside our control—we cannot do anything to improve it by changing the course design, and the students cannot do anything about it by changing their attitude in the CS1 course. We would like to identify success factors within our control, i.e. success factors that we can promote by changing aspects of our course design.

From our experience, we conjecture that other factors also are likely to be indicators of success than the ones investigated in the research reported in this paper. Numerous other factors might be success indicators in an introductory model-driven programming course, e.g. motivation, effort, power of abstraction, prior programming experience, social course context, emotional and social health, family background, ethnic background, financial situation, and computer literacy. In order to improve the learning situation, we would like to pursue those factors we believe to be dominant in predicting success and which we can do something about by changing our course design, and this rules out family background, ethnic background, financial situation and computing literacy.

Motivation. How motivated is the student? Presumably, a CS major is more motivated than a math major or chemistry major; and of course some CS students are more motivated than others.

Effort. How hard does the student work with the subject during the quarter/semester? Programming is a contact-sport, and the hard-working students are likely to perform vastly better than the less hard-working students are. In this research, we have used a simplified description of the effort the students puts in the course namely the result of the mandatory assignments.

Power of abstraction. We believe that the student's power of abstraction —the students ability to cope with abstract concepts and their detailed realization in a modern programming language which is a task spanning several orders of magnitude— plays a dominant role as indicator of success in any introductor programming course, also a model-driven course as ours.

Prior programming experience. All other things being equal, we expect prior programming experience to be an indicator of success. However, one often sees that students with prior programming experience rely too much on their prior experience and eventually find themselves (lost) far behind the students that approach their study with a more humble and hard-working attitude. This indicator has been shown in several studies [13, 30] to be an indicator of success, even though Ventura [28] could not confirm this.

Social course context. We believe the social context of the learning environment, i.e. the lecturer, the TA, and the fellow students, to be an indicator of success; however, it is probably a more moderate success indicator than the other four mentioned above.

Emotional and social health. [21] found that "both emotional and social health factors related to student performance and retention" (p24). In their study they have used a wide range of tests to determine college students emotional and social health. It would be interesting to see if these factors have the same impact on students participating in a model-based programming course.

It is not a trivial task to measure the parameters mentioned above; consequently, a major part of the indicated future work will be to identify trustworthy techniques of establishing quantitative measures of these parameters.

In section 5.3 we concluded: "In order to answer the more general question of success factors of the object-first CS1 course we need data from various different implementations of this teaching strategy." Therefore, we would like to extend the investigation to other institutions (other teachers, other interpretations of objects-first, etc.).

7. CONCLUSIONS

We have studied eight potential indicators of success for a model-driven CS1 course at university level: math grade from high school, course work, study age, major in CS, major in math, major in geology, major in nano science, and gender.

We have explained our specific interpretation of objects-first by presenting a detailed description of the course design including goal, form, content, exam, and pedagogy.

We have presented our research method including research question, data, statistical method (multiple regression analysis).

Of the eight potential indicators of success, we have found only two to be significant at a 95% confidence interval: math grade from high school and course work. The two significant indicators explain 24.2% of the variation of the exam grade. Math is the more dominant of the two, it accounts for 2/3 of the variation. The result concerning math grade contradicts the findings of Ventura [27].

We have discussed four aspects of our research:

1. The explanation power of the variables: the variables considered here definitely have a large impact on the result of the exam.
2. The impact of our findings on teaching: the significance of the course work variable indicates, not surprisingly, that students

who follow the pace of the course performs better at the final exam.

3. Limits of what to conclude from the available data: data from various different implementations of this teaching strategy is needed in order to answer the more general question of success factors of an object-first CS1 course.
4. The variety of the notion “objects-first”: because of the variety of interpretations of “objects-first”, it is impossible to make conclusions about this approach in general.

Further work need to be done in order to make generalizable results on the success factors for objects-first programming; we suggest six potential indicators of success that we believe to be dominant in predicting success and which we can do something about by changing our course design.

8. ACKNOWLEDGEMENT

We would like to thank Louise Heltborg for her help in the statistical analysis and Lisbeth Petersen for her help in collecting the data. We would like to thank all the participants in the course for taking part in our investigations.

9. REFERENCES

[1] 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.

[2] Bennedsen J. & Caspersen, M.E (2005). Revealing the Programming Process. Proceedings of the thirty-sixth SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, 2005. pp.186-190.

[3] Bergin, J. 14 Pedagogical Patterns. Available on-line at “http://csis.pace.edu/~bergin/PedPat1.3.html”. Last accessed May 13 2005.

[4] 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.

[5] Brooks, J. H., & DuBois, D. L. (1995). Individual and environmental predictors of adjustment during the first year of college. Journal of College Student Development, 36, pp. 347-360.

[6] 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.

[7] Caspersen, M.E. & Christensen, H.B. (2000). Here, There and Everywhere – On the Recurring Use of Turtle Graphics in CS1. Proceedings of the Fourth Australasian Computing Education Conference, ACE 2000 Melbourne, Australia, 2000, pp. 34-40.

[8] Caspersen, M.E. & Christensen, H.B.(2002) Frameworks in CS1 -- a Different Way of Introducing Event-driven Programming. In: Proceedings of the seventh Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2002, Aarhus, Denmark.

[9] Cooper, S., Dann, W.& Pausch, R. (2003) Teaching Objects-first In Introductory Computer Science SIGCSE’03 February 19-23, 2003, Reno, Nevada, USA. pp. 85 – 89.

[10] Exam score (2005). http://www.retsinfo.dk/_GETDOCM_/ ACCN/B19950051305-REGL (English translation can be found in the bottom) Last accessed April 30 2005.

[11] 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.

[12] Foster, T. R. (1998). A comparative study of the study skills, self-concept, academic achievement and adjustment to college of freshman intercollegiate athletes and nonathletes. Dissertation Abstracts International Section A: Humanities and Social Sciences, 58(12-A), pp. 4565.

[13] 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.

[14] The Joint Task Force on Computing Curricula (IEEE Computer Society and Association for Computing Machinery). Computing Curricula 2001 (final report), December 2001. Available on-line at “http://www.computer.org/education/cc2001/final”. Last accessed May 13, 2005.

[15] Jones, R., Boyle, T. & Pickard, P. (2003) Objectworld: Helping Novice Programmers to Succed through a Graphical Objects-first Approach. Proceedings of 4th Annual LTSN-ICS Conference, NUI Galway, pp. 111 – 114.

[16] Kölling, M. & Rosenberg, M. (2001) Guidelines for teaching object orientation with Java, ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education, pp. 33 – 36.

[17] 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.

[18] Lewis, J. (2000) Myths about object-orientation and its pedagogy, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, pp. 245-249.

[19] Nielsen, K., Kvale, S. (1997). Current issues of apprenticeship”. Nordisk Pedagogik, Vol 17, pp. 130-139.

[20] Nowaczyk, R. H. (1983). Cognitive skills needed in computer programming. Paper presented at the Annual Meeting of the Southeastern Psychological Association, Atlanta, Georgia.

[21] Pritchard, M. E. & Wilson G S. (2003). Using Emotional and Social Factors to Predict Student Success, Journal of College Student Development, Vol 44(1).

[22] 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.

[23] Sanders, R. T., Jr. (1998). Intellectual and psychosocial predictors of success in the college transition: A multiethnic

study of freshman students on a predominantly White campus. Dissertation Abstracts International Section B: The Sciences and Engineering, 58(10-B), pp. 5655.

[24] Schmolitzky, A. (2004) Objects first, interfaces next" or interfaces before inheritance, Educators symposium, OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications pp. 64 – 67.

[25] Szulecka, T. K., Springett, N. R., & de Pauw, K. W. (1987). General health, psychiatric vulnerability and withdrawal from university in first-year undergraduates. British Journal of Guidance & Counseling Special Issue: Counseling and health, 15, pp. 82-91.

[26] 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.

[27] 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.

[28] 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, Volume 36(1) pp. 240 – 244.

[29] Wang, M. C. & Haertel, G. D. (1993). What helps students learn? Educational Leadership; Dec93/Jan94, Vol. 51 Issue 4, pp. 74-79.

[30] 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