Success factors in an introductory programming course in a non-CS major

MAIN FIELD: Informatics
AUTHOR: Tilda Alnervik & Gerui Ma
SUPERVISOR: Johan Kohlin
JÖNKÖPING 2020 March
This final thesis has been carried out at the School of Engineering at Jönköping University within informatics. The authors are responsible for the presented opinions, conclusions and results.

Examiner: Ulf Seigerroth
Supervisor: Johan Kohlin
Scope: 15 hp
Date: 2020-03-08
Abstract

Purpose – The contradiction of the increased demand for IT specialists and the decrease of the enrollment in programming courses at universities worldwide has been discussed over the years. To tackle the problem, researchers and teachers in computing education have investigated various success factors in introductory programming courses, mostly within the context of computer science. This paper focuses on the investigation of success factors for students that are not majoring in computer science (non-CS) in an introductory programming course, to report on the results of student’s performance and analysis of the most relevant success factors, also provide suggestions that could be considered for the course design and teaching method.

Method – The methods used to carry out the study are a survey with 36 participants, conducted before the start of an introductory programming course, and qualitative interviews conducted with twelve students after the end of the course. The interviews were then analysed thematically to find common patterns for five success factors between the students with different grades. The success factors that were examined are math background, previous programming experience, comfort level, motivation and attribution to success.

Findings – Math background could not be proved as a success factor in this study due to the lack of a standardized assessment of the students’ math levels. Previous programming experience could be regarded as a success factor but not as dominant as the success factor motivation, which has shown clear patterns in the data. Comfort level could be seen as one of the success factors as well, as most of the data in this study support this conclusion. Lastly, attribution to success as a success factor could not be supported by this study; the qualitative data showed variety which makes it hard to draw a conclusion directly.

Implications – The study suggest increasing the motivation for the non-CS students in introductory programming by combining programming with other subjects in the programme. The lecturer could customize the course for students with different interests so they could select a path and adapt the knowledge to their needs. Bridging courses and various forms of mentoring are also recommended to offer.

Limitations – The time frame of the study limited the amount of data that could be collected. The study was conducted with students from only one university and one non-CS programme, with a small data sample for analysis, which is limiting in the way the results can be generalized.

Keywords – introductory programming, non-CS major, informatics, success factors, math background, previous programming experience, motivation, comfort level, attribution to success
# Table of Contents

**Abstract** ................................................................................................................................. ii

**Table of Contents** ..................................................................................................................... iii

## 1 Introduction ............................................................................................................................ 1

1.1 BACKGROUND ..................................................................................................................... 1

1.2 PROBLEM STATEMENT ....................................................................................................... 1

1.3 PURPOSE AND RESEARCH QUESTIONS ......................................................................... 2

1.4 THE SCOPE AND DELIMITATIONS ................................................................................ 2

1.5 DISPOSITION ....................................................................................................................... 3

## 2 Method and implementation .................................................................................................. 4

2.1 LINK BETWEEN RESEARCH QUESTIONS AND METHODS ........................................... 4

2.2 WORK PROCESS AND RESEARCH DESIGN ................................................................. 4

2.3 APPROACH ....................................................................................................................... 4

2.4 DATA COLLECTION .......................................................................................................... 4

2.4.1 Survey ......................................................................................................................... 5

2.4.2 Interviews ................................................................................................................... 5

2.4.3 Literature search ....................................................................................................... 5

2.5 DATA ANALYSIS .............................................................................................................. 6

2.5.1 Survey ......................................................................................................................... 6

2.5.2 Interviews ................................................................................................................ 6

2.6 VALIDITY AND RELIABILITY .......................................................................................... 6

## 3 Theoretical framework ......................................................................................................... 8

3.1 LINK BETWEEN RESEARCH QUESTIONS AND THEORY ........................................... 8

3.2 INTRODUCTORY PROGRAMMING COURSE ............................................................... 8

3.3 SUCCESS FACTORS ......................................................................................................... 8

3.3.1 Math Background ..................................................................................................... 9

3.3.2 Previous Programming Experience ........................................................................... 9

3.3.3 Comfort Level ......................................................................................................... 10

3.3.4 Motivation ............................................................................................................... 10

3.3.5 Attribution to Success ............................................................................................ 11

3.4 TEACHING METHODS ...................................................................................................... 11

3.5 COGNITION STYLES IN CONNECTION WITH PROGRAMMING SKILLS .................. 11
4 Empirical research

4.1 COURSE INTRODUCTION AND GRADING CRITERIA

4.2 SURVEY

4.2.1 Final Exam Statistic
4.2.2 Math Background
4.2.3 Previous Programming Experience
4.2.4 Motivation
4.2.5 Demographic Question

4.3 INTERVIEWS

4.3.1 Math Background
4.3.2 Previous Programming Experience
4.3.3 Comfort Level
4.3.4 Motivation
4.3.5 Attribution to Success
4.3.6 Demographic Statistic of the Interviewees
4.3.7 Observations of the language and wording used by interviewees

5 Analysis

5.1 HOW DOES MATH BACKGROUND CONTRIBUTE TO SUCCESS IN INTRODUCTORY PROGRAMMING IN A NON-CS MAJOR?

5.2 HOW DOES PREVIOUS PROGRAMMING EXPERIENCE CONTRIBUTE TO SUCCESS IN INTRODUCTORY PROGRAMMING IN A NON-CS MAJOR?

5.3 HOW DOES COMFORT LEVEL CONTRIBUTE TO SUCCESS IN INTRODUCTORY PROGRAMMING IN A NON-CS MAJOR?

5.4 HOW DOES MOTIVATION CONTRIBUTE TO SUCCESS IN INTRODUCTORY PROGRAMMING IN A NON-CS MAJOR?

5.5 HOW DOES ATTRIBUTION TO SUCCESS CONTRIBUTE TO SUCCESS IN INTRODUCTORY PROGRAMMING IN A NON-CS MAJOR?

6 Discussion and conclusion

6.1 FINDINGS
6.2 IMPLICATIONS
6.3 LIMITATIONS
6.4 CONCLUSION
6.5 FURTHER RESEARCH

References

Appendices
1 Introduction
The chapter provides a background for the study and the problem area the study is built upon. Further, the purpose and the research questions are presented. The scope and delimitations of the study are also described. Lastly, the disposition of the thesis is outlined.

1.1 Background
In the past years, the demand for programmers has been increasing (Vikute-Adzgauskiene & Vidszunas, 2012). Students like to live a life influenced by computing, and many want to work within the field after their education (V. Barr & C, 2011). Students also used to show more interests in the programming courses, not only within the discipline of computer science but also for the non-CS majors (Robins et al., 2003; Dawson et al., 2018). However, programming courses are generally regarded as difficult, and usually have high dropout rates (up to 50%) which have led to the decrease of programming courses’ enrollment nowadays (De Raadt, 2007; Guzdial, 2003; Rich Perry, & Guzdial, 2004). Universities find it harder to attract students without lowering the entry scores (Ford. M & Venema. S, 2010).

Furthermore, students’ programming skills after completing these courses are often evaluated as insufficient (Nikula et al., 2007). In 2001, the McCracken research group assessed first-year computer science students to investigate their programming skills after taking one or two introductory courses. Two hundred sixteen students from four universities performed poorly with an average score 22.89 out of 110, even though the research group considered the test questions to not be difficult and that “students in any type of Computer Science programme should be able to solve them” (McCracken et al., 2001). Ma et al. (2007) expressed concern that approximately one-third of the students did not understand the fundamentals of programming after an introductory programming course.

Students face diverse problems when they learn to program, while non-CS majored students tend to show even less contentment with their experience and lower grades (Dawson et al., 2018). Universities thus must adjust their introductory programming courses to avoid high dropout rates, meanwhile ensuring that the students learn what they are expected to (Ford. M & Venema. S, 2010).

The situation has led to various attempts to solve the problems. Ford and Venema (2010) took inspiration from the McCracken et al. (2001) and Ma et al. (2007)’s assessments of the programming students to evaluate the success of introductory programming courses. Robins et al. (2003) defined the effectiveness and ineffectiveness of novice programmers and provided suggestions to novice programmers on how to improve their strategies of problem-solving. Kasto (2016) did research on the learning processes of novice programmers to support the enhancement of programming education. Dawson et al. (2018) developed an introductory programming course CS0.5 (in comparison to CS1) to meet the needs of the diverse population of non-CS major students interested in the course. Hausswolff & Eckerdal (2018) describes the process of developing a context-specific assessment tool for novice programmers in their classes.

1.2 Problem Statement
Why is programming so hard to learn? How can the introductory courses for programming be improved for beginners? Previous studies have provided theoretical explanations and practical solutions to programming learning problems and obstacles. Nevertheless, a lot of research about programming learning is based on a broader computer science context. For example, the McCracken (2001) working group conducted their empirical study of novice programming assessment with students who studied in their first year of computer science, and the study aimed to provide an assessment to the computer science community. Kasto (2016) did a longitudinal study that followed a few students through one academic year of learning to program at Auckland University of Technology (AUT). Those studies did not specify the non-CS majors as a group in the participants. Dawson et al. (2018) at the University of British Columbia have however identified the similarities and differences between the CS-major and non-major students’ experiences and course outcomes, in order to develop an introductory programming course “CS0.5” to the non-majors.
The purpose of our study is like Dawson et al. (2018) but will be having different approaches and types of outcomes. Our interest in this broad topic is emphasized on non-CS major students in introductory programming courses focusing on students in the introductory course Client-Side Programming in the programme New Media Design at Jönköping University in Sweden. The programme has a major in informatics and provides graphic design, UX design, and web development courses, but many of the students during the past three years have experienced a hard time with programming courses and tended to identify themselves as designers who know programming instead of the opposite.

1.3 Purpose and research questions
Drawing on the problem statement, it is evident that there is a lack of studies on success factors in introductory programming in a non-CS major. Consequently, the purpose of this study:

Is to evaluate success factors in introductory programming in a non-CS major.

Five success factors were chosen based on a list by Sharma & Shen (2018) of 37 factors that has been studied in the past. Success factors were excluded due to for example not being applicable for qualitative interviews which is the chosen method for this study, like cognitive and behavioral tests and testing different teaching methods. Others were excluded because of being very similar, where only one would be chosen. Also, some success factors were excluded because of the limitation of respondents in the interviews. For example, some success factors required having knowledge about students’ educational background which would be difficult in the international programme where this study will be conducted. The five chosen success factors were all regarded as significant in previous research within computer science majors. We hope to bring those previously studied factors one step further to a non-CS major context with our work, as there are multiple university-level programmes in Sweden that are not Computer Science, but do have programming courses, for example, New Media Design at Jönköping University, Interaction Design at Malmö University, User Experience Design at Skövde University and IT-Design – System Design at Karlstad University to name a few.

The planning and execution of introductory programming courses can be facilitated with a better understanding of students and their backgrounds and abilities, which this study is believed to give. We hope to provide some suggestions to both the teachers and the non-CS major students in such courses, and further research could move on by utilizing some of the influential factors in their experiments or examine them with different approaches.

To be able to fulfil the purpose, it has been broken down into five questions concerning the different success factors.

[1] How does math background contribute to success in introductory programming in a non-CS major?

[2] How does previous programming experience contribute to success in introductory programming in a non-CS major?

[3] How does comfort level contribute to success in introductory programming in a non-CS major?

[4] How does motivation contribute to success in introductory programming in a non-CS major?

[5] How does attribution to success contribute to success in introductory programming in a non-CS major?

To answer the questions and thereby to fulfil the purpose, explanatory research will be conducted.

1.4 The scope and delimitations
The study has a limitation on its time and scope that the participants are students who enrolled in one introductory programming course at one university, which is relied on single snapshots
of student work. The time scope also limited the study with the data collection that only 36 students participated in a survey. A few consequences can be caused by these limitations. Primarily, such small sample size in the quantitative research might prevent the findings from being extrapolated well. Ideally the study could achieve higher validity if some students from other non-CS programmes that enroll in introductory programming courses could participate in the study. However, the study decides to take another approach to be more effective within the limited time frame, which is to use a research method called triangulation. Basically, the study utilizes multiple methods to support the validity, qualitative research is done in the study to compensate these limitations, twelve students participated in the semi-structured interviews to answer the thematic questions in-depth. The next chapter will explain more about the research methods.

In the interviews, students were asked about their final exam result of an introductory programming course “Client-Side Programming” at Jönköping University; however, the answers could be not fully reliable since their personal grades will not be checked through the official education system Ladok.

Due to time limitation, the study will not be profoundly investigating different university programs and courses. The factor of teaching methods will be considered and discussed only in the literature survey, as the data collection has a limited scope within one university and one course, where all the students in the spectrum of grades have had the same teaching methods in the introductory programming course. The teaching methods themselves will not be taken into account for the success factors.

Besides, the study will not examine how to use the results of the success factors found practically.

1.5 Disposition
The thesis is structured by first describing the method and implementation used to gather and analyze empirical data in chapter two. Chapter three contains the theoretical framework that covers previous research in the field and works as a foundation for the study. Thereafter follows chapter four, focusing on the empirical data that was gathered, and in chapter five, an analysis is presented, answering the research questions. Chapter six concludes the study with a discussion of the findings, limitations and further research. After these chapters, a list of references and appendices is available.
2 Method and implementation

The chapter provides an overview of the work process of the study. Further, the approach and design of the study are described as well as the data collection and data analysis. The chapter ends with a discussion about the validity and reliability of the study.

2.1 Link between research questions and methods

The following chapter describes the chosen methods for data collection and data analysis, as well as how they answer the research questions.

Research questions 1, 2 and 4 will be partly answered through a survey. Research question 3 and 5 require that the student had taken the final exam, which was not the case when the survey was conducted.

All five research questions will be covered in the qualitative interviews.

2.2 Work process and research design

To be able to understand the problem stated in the previous chapter, a literature study was conducted. The literature was used to set a foundation for the study and find information on the previous research on introductory programming. This is the part where we found information about success factors that have previously been studied and decided on how to structure our study further.

Much of the previous research was made with a quantitative research design. We decided to incorporate that into our study as well, through creating a survey that the students answered in the very beginning of the introductory programming course. But the previous research had a very large group of respondents on their surveys, making it easier to generalize the results. Some of these studies were also conducted over multiple semesters (Ventura, 2005) or multiple countries (Sharma & Shen, 2018), making the reliability of the studies higher. With the limited scope and time frame, this study did not have the opportunity to collect enough data for a solid quantitative analysis, which is why a qualitative research method is added.

A qualitative method is suitable for when the motive is to develop a deeper understanding of a phenomenon, and when interest is found in ambiguity (Blomkvist & Hallin, 2014). The study’s interest lay in the understanding of success in introductory programming and through a qualitative approach, namely interviews, which supports a nuanced comprehension of what makes students perform on different levels through a student’s perspective.

2.3 Approach

The study follows a deductive approach, meaning that the hypotheses are formed through previous research on the topic. Through the literature, the study found several factors that had been demonstrated to be predictors of success in introductory programming, and the hypothesis became that these factors would also show to be significant in a non-CS major. However, the analysis will also be inductive, meaning that the findings will be based on what has been found in the interviews and trying to find patterns that are generalizable and not be bound to the results of previous research.

2.4 Data Collection

The empirical data was collected with two separate methods, a survey and interviews. Using more than one method to collect data is called triangulation and is used to achieve higher reliability (Patel & Davidson, 2019).

All the data was collected at Jönköping University, in the programme New Media Design, specifically in the course Client-Side Programming, which is the students’ first course in programming in the programme. The course uses a structure of first having theoretical lectures, which are followed up by the teacher showing how to solve small tasks with the theories from lectures, and lastly, the students get lab assignments based on the content of the lecture. During the labs, students can get individual help with the assignments from the lecturer and/or lab assistants that are generally students from the year above. The teaching method of both letting
the students watch the lecturer code and letting them solve their own assignments conform with the suggestions on by Kölling and Barnes (2004) on using apprentice learning as well as problem-based learning in introductory programming.

2.4.1 Survey
During autumn of 2019, in the beginning of the course Client-Side Programming I, a survey of 13 dichotomous questions was created. The survey focused on different success factors that were possible to gather information on before the students had started the course, namely background, previous programming experience and motivation. The survey was conducted during the student’s second lecture in the course. Out of 65 students registered in the course, 36 participated in the survey. The survey was anonymous if the student wanted, however, a field for input of contact information was added for students who were willing to later participate in interviews.

2.4.2 Interviews
The main source of data in the thesis was collected through semi-structured interviews with a low degree of structure and a high degree of standardization. The low degree of structure allows the respondent to freely answer the questions with their own words, and there are no predetermined answers (Trost, 2010). When freedom is given during the interviews, the interviewees could be able to express themselves more openly which benefits the validity of the study. The high degree of standardization, however, means that the questions were the same for all respondents and asked in the same order (Patel & Davidson, 2019). This was done to be able to generalize and compare the answers from the different respondents. With the high degree of standardization, the interview conductor could prepare the thematic questions beforehand which means that the interview will still be looked after with certain outlines so that the informants will not go off from the thematic topics.

With the limited time frame of the bachelor thesis, only a small number of students that had taken the course Client-Side Programming were interviewed. The students that participated in the interviews had either voluntarily submitted their contact information during the initial survey or were asked to participate directly by the authors. The students were in either year one or two of the programme. They all had in common that the Client-Side Programming course was the only programming course they had taken so far during the bachelor programme.

The interviews were conducted face-to-face with the respondents in the university to create a setting where they felt comfortable. Since the bachelor programme to which the Client-Side Programming course belongs is international, all interviews were held in English, which is neither the interviewer nor the respondents’ first language. This may have influenced/and or limited the wording of the answers.

The questions asked in the interviews were grouped into different themes based on the success factors we wanted to explore. There were also some questions that were designed to catch other success factors than the ones we are primarily investigating. In total, 22 questions were asked, and the interviews lasted around 15-30 minutes for each respondent.

2.4.3 Literature search
As for the literature search, a library platform called Primo at Jönköping University (JU)’s website is used. On this platform, books, articles, and other publications could be found in printed or electronic form. A few literatures were also borrowed from the JU library in physical form. When searching for literatures on Primo, a filter “peer reviewed article” is often used to secure the quality of the paper. Search key words were selected based on the thesis topic and combined with each other, for instance “introductory programming”, “non-CS programming”, “success factors programming”, “success factors introductory programming”, “non-CS course”, etc.
2.5 Data Analysis

The final exam in the Client-Side Programming Course is an exam that is partly theoretical and partly practical and is supposed to cover all the intended learning outcomes in the course. For this reason, we decided to base our analysis on the students' grade on the final exam. A student's grade has been used previously to determine success (Wilson, 2002; Gomes & Mendes, 2010; Ayalew et al., 2018). With the limited scope of the bachelor thesis, we only had the opportunity to interview 12 students. The grades of these students were the following:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>U (Fail)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Interviewees' grades on the final exam

The few numbers of students interviewed with grade Fail and 4 made us divide the participants into two groups, low grades – which includes Fail and 3, and high grades – which includes grade 4 and 5.

In this thesis, we will compare the different grade groups with the answers the students within those groups have given to hopefully find some patterns and be able to answer the research questions.

2.5.1 Survey

Since the survey was anonymous, it was not possible to compare a student's answers to their grade on the final exam. Instead, the survey was analyzed by comparing the class' average results with the results of the survey. Most questions were dichotomous, meaning they were yes or no questions, which made it easier to compare percentages of a specific answer in the survey to percentages of different grades in the class.

2.5.2 Interviews

Twelve interviews with students were conducted. The audio was recorded after getting consent from the students and later transcribed by the interviewers. The interviews were analyzed thematically, which means that the empirical data was sorted under different categories. The categories are then used to be able to answer the research questions (Blomqvist & Hallin, 2014). The categories used can, for example, be found when going through the collected interview data, or it can be categories found in previous research in the area (Blomqvist & Hallin, 2014). The study started with categories found in previous research – namely the five success factors that will be examined. Subsequently, the interviewees' answers under the different categories were grouped and looked for patterns within the groups of students with low grades as well as high grades. This will be done by comparing the answers from the two groups and examine if one group has some distinct characteristics on most of the students, while the other group does not have. Namely, differences and similarities between the two groups will be analyzed based on the success factors. However, the study had an open mind for trying to discover other themes than the ones initially stated as well while analyzing the collected data.

2.6 Validity and reliability

Trost (2010) argues that talking about reliability in a qualitative study seems peculiar since it requires measuring of static variables which do not exist in the qualitative study. However, there are ways to consider increasing reliability, even in this type of study. For example, objectivity, which is one of the components of reliability (Trost, 2010), can in the case of this thesis be increased by reviewing the collected interview data multiple times and through discussion between the authors to make sure that the interview answers cannot be interpreted differently by different readers.

Another component of reliability is congruence, which signifies the similarity between questions that measures the same thing (Trost, 2010). It is something that we achieved by asking multiple questions concerning the success factors. That way, it makes it easier to see different nuances of the same question.
The reason for doing the interviews in a standardized way, e.g. the same environment, same questions in the same order etc. was to achieve higher reliability as we compare the respondent’s different answers.

Another important part that we believe increases the reliability in the study is the literature study. Through reviewing work by others and the results that have been discovered by these, and comparing them to the study we conduct, we can get an understanding of the reliability of our work.

The combination of two research methods, so-called triangulation (Patel & Davidson, 2019) is a research method that further increases the reliability of the study. Through comparing the different results of the survey and the interview, we get a better understanding of which success factors that are significant in the course.

The validity concerns if the question measures what is supposed to (Trost, 2010). However, the collection of data is not the only thing concerning validity with a qualitative method, but it concerns the whole research process (Patel & Davidson, 2019). Patel and Davidson (2019) further argue that an important part of validity is how the data is interpreted and if the results add knowledge within the studied area.
3 Theoretical framework

The chapter presents the theoretical foundation for the study.

3.1 Link between research questions and theory

The following chapter covers the theoretical foundation needed for answering the research questions.

This study began by seeing a problem in our own courses where many students would fail or drop out because of the difficulty in the introductory programming course. While starting to do a literature survey, it was evident that the problem of introductory programming is worldwide known. Continuing the literature study, research was found that explored different success factors and through the success factors that seemed to have the most impact on results in introductory programming in CS-majors, the research questions for this study were designed.

In the following sections, introductory programming and the five success factors are explored further.

3.2 Introductory Programming Course

Introductory programming is a phrase that will be used frequently throughout this thesis. The phrase in this context refers to a course on a university level, where the students are taught programming. The course does not have any requirements from the student of having previous knowledge in the programming language or even programming in general for taking the course.

In much of the research from the literature study, the research has been done on introductory programming courses are called CS1, which stands for Computer Science 1, and they are usually the first course in computer science major where students learn programming. However, there are courses teaching programming outside of CS majors, and that is the main interest of this thesis.

3.3 Success factors

Predicting success and finding success factors for introductory programming courses has been studied for about 40 years since the IT industry and programming education started taking off. Many different factors for success have been researched upon, in fact, Sharma and Shen (2018) compiled a list of 37 different factors that had been studied in the past by the time that study was conducted. Many of the factors that has previously been studied include different instruments and behavioral and cognitive tests that were not applicable due to the chosen method of qualitative interviews in this study. Furthermore, some success factors were hard to study because of the limited data collection. The students enrolled in the program New Media Design who were to be interviewed are from different countries around the world, so to include for example “previous academic experience” and “prior composite academic ability” would be difficult since an understanding of each students’ previous studies and levels from different school systems would have to be considered. Other factors that has been excluded from the study are factors regarding work styles, learning approaches, learning styles, medium of instruction as these are all constants for the students participating in the study. To evaluate these, the study would have to span multiple semesters or be conducted with students from different universities where the teaching styles vary – to be able to compare success between students who have been exposed to different learning styles etc. In this study some similar success factors were excluded due to the timeframe, for example among the factors “previous programming experience”, “previous computing experience”, “previous non-programming computer experience” and “experience on the module” only “previous programming experience” was chosen. Also “encouragement from others” and “keeness and general academic motivation” were combined into the success factor Motivation. Gender is a success factor that has been studied in the past and would have been interesting to study but would have required a bigger data sample than possible to gain in the scope of this thesis, as there would have to be multiple students of each gender in each group of low grades and high grades. Of course, there may also exist success factors that were not in the list by Sharma & Shen (2018) that we have not considered.
The five success factors that were chosen for this study, math background, previous programming experience, comfort level, motivation and attribution to success have all been regarded as significant in previous research. We also initially thought that they would all fit our chosen methodology in contrast to many of the other success factors mentioned by Sharma and Shen (2018). Since previous research has been made in courses for CS-majors, the interest in success factors in this study is if they are significant in a non-CS major as well.

Below information and discussion about the five factors that will be examined in this study is found.

3.3.1 Math Background
Math background as a predictor of success in introductory programming has been widely studied. More than 30 years ago Campbell and McCabe (1984) wrote about the relationship between math background and the persistence of a freshman in a computer science major and came to the conclusion that the students who persisted in a major in computer science had taken more semesters of math in high school, had higher math grades and higher scores on SAT math than the students who did not persist in a computer science major.

About a decade later, Wilson (2002) made a study on factors promoting success in an introductory computer science course with one of the factors studied being math background. Similarly to the study by Campbell and McCabe (1984), Wilson used number of semesters of math courses taken by the student in high school as a way of defining math background and found that math background was the second most important factor to determine success and that it had a positive influence on the results on the course.

Besides, Bergin and Reilly (2005) found a relationship between previous academic experience in mathematics and performance on a first-year programming course at a university, with it being in the top three strongest correlations to performance out of the 15 factors studied.

Gomes and Mendes (2010) also studied student performance, and programming teachers they talked to during the study claimed that one reason for students struggling with programming has to do with the insufficient mathematical background to be able to solve problems.

A study made on factors affecting programming performance on first-year students in more recent years was made by Ayalew, Tshukudu & Lefoane (2018) on a university in Botswana. Just like the other studies mentioned in this section, they found a correlation between high school performance in mathematics and the performance in an introductory programming course.

By analyzing the research that has been made on math background as a predictor of success in introductory computer science/programming, it is safe to say that math background is one of the most important factors of success in introductory programming.

3.3.2 Previous Programming Experience
It may be easy to form a hypothesis that previous programming experience will have a positive impact on the results on an introductory programming course, but during the literature survey, different results were found. Some of the research says previous programming experience does have a correlation to performance in an introductory programming course, and some have not found these correlations.

For example, Ayalew et al. (2018) investigated the relationship between performance in an introductory programming course and the students’ performance in high school computer studies where programming was taught and did not find any correlations. Similar results have also been found in research by Bergin and Reilly (2005) and Ventura (2005).

In contrast to the studies mentioned above, Watson, Li and Godwin (2014) found a significant difference in performance between students having previous programming experience and those who did not. However, when researching whether there is a correlation between performance in the course and number of years of programming experience and number of programming languages the student had experience in, the researchers did not find any significant difference between the students. Other studies with similar results include Wilson
(2002), who found that previous programming experience had correlations to the performance on the midterm exam in an introductory computer science course. However, she did not find the previous experience to be a significant predictor of success on the whole course.

In 2018, a study was conducted including previous programming experience as a success factor, comparing the results of two different universities in India and Australia (Sharma and Shen, 2018). The results showed that students who had self-initiated studies in programming before the course started performed better in the introductory course at the Australian university, but for the Indian university the result showed no correlations between the factor and student performance.

Wilcox and Lionelle (2018) did a study on an introductory programming course, focusing only on how precious experience influenced the performance. Their results showed that students with previous knowledge performed significantly better in the course. They did, however, continue their study onto the next programming course and saw that the gap in performance had been closed between the groups.

3.3.3 Comfort Level
Comfort level is the students’ perception of comfort in the course relating to asking and answering questions on lectures, asking questions in labs, asking questions during office hours, perceived anxiety and difficulty while writing programming code and solving assignments and perceived understanding of programming compared to classmates and was first studied by Wilson (2002).

Comfort level as a predictor of success in introductory computer science has been studied various times with similar results. In her study of 12 success factors in an introductory programming course, Wilson (2002) found comfort level to be the most important predictor of success. Bergin & Reilly (2005) found comfort level to be a significant predictor of success combined with the factors math background, perception of the understanding of the subjects taught, and gender. However, the research showed students’ perception of their understanding to be the crucial predictor of success. In another study, Ventura (2005) also found comfort level to be the second strongest predictor of success in an introductory programming course, after student effort.

Wilson (2002) remarks that it was interesting to see in her research that comfort level was found to be more important than math background, which according to much research is seen as the top predictor of success in introductory computer science.

3.3.4 Motivation
Another interesting factor that may or may not affect the student performance is the motivation for studying programming. Does the reason behind choosing to study programming correlate with the student’s results?

In Sharma and Shen’s (2018) comparative study between two universities, they found that students who said that they were not interested in learning programming scored lower than the students who did say that they were interested at the Australian university, suggesting motivation to be a predictor of success. However, at the Indian university in the same study, they did not find any correlations between the exam score and the reasons to study programming.

Zainal et al. (2011) conducted research on students’ perception and motivation in an introductory programming course. They divided motivation into four types (using a model by Jenkins, 2001), being extrinsic, intrinsic, social and achievement. Extrinsic motivation in this context is, for example, that the student study programming to have a successful career or to influence success in other ways. Intrinsic motivation means that the student study programming because of interest. The study showed that “Students who have extrinsic motivation are those who are categorized in excellent and good achievement” (Zainal et al., 2011) suggesting that students who study programming because of interest perform better. But also, students with intrinsic motivation “show positive behaviour and attitudes to achieve excellent results”.

---

10
3.3.5 Attribution to Success

Attribution theory is a model to explain the process by which individuals explain the causes of events of behaviour. In the context of this study, attribution will be looked upon as the cause student’s give to their success or failure in an introductory programming course. In other words, why does the student think they succeeded or failed in the course, and is the attribution a factor in predicting a student’s performance?

The original attribution theory includes four different types of attribution to success and failure (Hawi, 2010). They are attribution to ability - the student perceives their performance to be a cause of their own ability to solve programming assignments, attribution to difficulty of task - the students perceive their performance to be a cause of the difficulty of the tasks they were given, attribution to luck - the students perceive their performance to be a cause of luck, and attribution to effort - the students perceive their performance to be a cause of their effort in for example studying in the course (Wilson, 2002).

Wilson (2002) found that attribution to success or failure to luck influenced the performance negatively, meaning that the students who thought their success or failure depended on luck were more likely to perform worse on a midterm exam than other students. Also, attribution to the difficulty of the task had a negative correlation to the results of the midterm exam.

3.4 Teaching Methods

The phenomenon of the decreasing popularity in programming courses in contradiction to the rapid IT revolution has been investigated by different authors (Vikute-Adzgauskiene & Vidzunas, 2012). Meyer (2003) state that the main reason for that is curriculum complexity and abstractness, its insufficient links with practical needs and the proliferated demand. Teaching method could be one of the factors to be considered when it comes to the incompleteness of programming courses.

ACM Computing Curriculum (2001) also known as the American Association for the Computing Machinery has provided well-known and influential recommendations of programming teaching strategies to universities until the last decade, which is based on the mathematical methodology of teaching (Dijkstra, 1997). The recommendations can be implemented by using different teaching methods, and the tendencies have been recognized by ACM Computing Curriculum (2008). One of the traditional teaching methods is based on imperative and functional paradigms, which could provide the students good theoretical introduction but often may lack knowledge on software engineering concepts for IT professionals (Vikute-Adzgauskiene & Vidzunas, 2012).

Teaching introductory programming at a university level has also been discussed by computer science teachers (Astrachan et al., 2005; Bailie et al., 2003; Bruce, 2004). Kölling and Barnes (2004) proposed apprentice-based and problem-based learning (PBL) in introductory programming courses. Bennedsen & Caspersen (2003) found that process recording is an efficient method to use in an introductory programming course compared to traditional classroom teaching. Nuutila et, al (2008) reported about experiments with tutorless PBL at Helsinki University of Technology (TKK) to test if the method could be implemented in the university courses and the result was not encouraging.

3.5 Cognition Styles in Connection with Programming Skills

Akinola et al. (2011) did qualitative research on student-programmers to investigate the connection between their cognition styles (Myers-Briggs Type Indicator (MBTI)) and programming skills. The Myers-Briggs Type Indicator (MBTI) is a test that indicates different personalities that influence people in decision-making and how they perceive the work, as the world’s most used personality assessment.

According to the Myers-Briggs company (2020), MBTI helps people understand their personality preferences in four key areas:

- How you get your energy (Extraversion vs. Introversion)
- How you take in information and learn (Sensing vs. Intuition)
• How you make decisions (Thinking vs. Feeling)
• How you like to organize your time and environment (Judging vs. Perceiving)

<table>
<thead>
<tr>
<th>Extraversion</th>
<th>Introversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing</td>
<td>iNtuition</td>
</tr>
<tr>
<td>Thinking</td>
<td>Feeling</td>
</tr>
<tr>
<td>Judging</td>
<td>Perceiving</td>
</tr>
</tbody>
</table>

Table 1: The Four Dichotomies of Myers-Briggs Type Indicator (MBTI) (Myers, 1962).

The result of Akinola et al. (2011)’s experiments showed that student-programmers with MBTI cognitive style ISTJ seemed to have better programming skill than the participants with other cognitive styles. The Myers-Briggs company (2020) defines ISTJ as “often clear and matter-of-fact, realistic, thorough, and they rarely miss details, tend to be calm, analytical, and objective.” Figure 1 presents the data from Akinola et al. (2011)’s study which indicates the dichotomies of MBTI of better skilled student programmers. Akinola et al. (2011) stated that the dichotomy of Extraversion/Introversion and Thinking/Feeling has little effect on the programming ability of an individual, with a minor difference, for example, Introversion seems to have a more positive influence on the individuals’ programming skills. The Sensing/Intuitive dichotomy showed the influence that most of the participants with the Introversion and Sensing dichotomy performed better in the programming assessment than the participants with the Thinking function (Akinola et al., 2011).

![Figure 1: Chart of the MBTI Dichotomy breakdown (Akinola et al., 2011).](image-url)
4 Empirical research

The chapter provides an overview of the empirical domain which forms the basis of this study. Further, a description is given of the empirical data that has been collected to answer the research questions.

4.1 Course Introduction and Grading Criteria

Client-side Programming I 7.5 Credits started in the autumn semester 2019 at Jönköping University. In this study, most of the data is collected from students who enrolled in this course, and a few of the students that are interviewed have participated in the same course one year earlier, with same course content but a minor adjustment in the lab assignments.

The course comprises modules giving theoretical as well as practical skills for developing modern web applications. The course includes the following parts:

- JavaScript, object-oriented client-side programming
- Manipulating HTML and Document Object Model
- Event-driven programming

The final grade for the course is based on a balanced set of assessments. The final grade will only be issued after satisfactory completion of all assessments (Client-side Programming I 7.5 Credits, 2019).

Grading system in descending order: 5=five, 4=four, 3=three, U = Fail.

<table>
<thead>
<tr>
<th>Name of the Test</th>
<th>Value</th>
<th>Grading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written examination</td>
<td>5 credits</td>
<td>5/4/3/U</td>
</tr>
<tr>
<td>Laboratory assignments</td>
<td>2.5 credits</td>
<td>5/4/3/U</td>
</tr>
</tbody>
</table>

Table 2: Registration of examination (Client-side Programming I 7.5 Credits, 2019)

4.2 Survey

Out of the 65 registered students in the course, the study did a survey with 36 of them before they took the written exam of Client-Side Programming. All the course statistics are provided by the lecturer of Client-side Programming, extracted from the education system Ladok without revealing any personal information of the students.

This section focuses on illustrating the empirical data collected from the survey. The questions are closed-ended with standardized answers and are largely dichotomous; therefore, graphs are used to illustrate the comparisons and percentages of the answers.

4.2.1 Final Exam Statistic

Fifty-eight students took the final exam, which is approximately 89% of total registered students. Table 3 presents the number of students under each grade criteria. To better illustrate the result with visual comparisons, figure 2 is used to indicate the percentage of each group according to the students’ exam results. It could be seen that more than half of the students who took the exam got the lowest grade 3, nine students failed, which ends up in the second largest group. A similar number of students got grade 4 and 5, which is a smaller group compared to the lower grade groups.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Number of Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>U (Fail)</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3: Statistics of exam result from Client-Side Programming I Autumn 2019
4.2.2 Math Background

In this section of the survey, students were asked to answer questions related to their math background. Figure 3 presents student’s enjoyment of math study in secondary school, 61% of them answered positively while 39% did not enjoy it. Figure 4 indicates the students’ comfort and confidence level of math, 33% answered yes to the statement “I found math easy and I’m a quick math learner”, and 67% had a negative response.

Question 1: I enjoyed study math in secondary school.

Question 2: I found math easy and I’m a quick math learner.
4.2.3 Previous Programming Experience

In this section of the survey, students were asked to answer questions related to their previous programming experience. Figure 5 presents the students’ perceived current level of programming knowledge when the course just began. On a scale 1-5 (low to high level), more than half of the students chose the lowest level, the other half of the students chose scale 2 and 3, no one had an answer with scale 4 and 5.

Figure 6 illustrates the number of students who took a programming course prior to the Client-Side Programming course. 75% clarified with “no” answer, in comparison to 25% who had prior programming courses before Client-Side Programming I. Figure 7 presents the result of a follow-up question which asked the students if they had done self-study on programming before the course started. The majority answered negatively; only 9% had such experience.

Figure 8 shows the students’ confidence level of programming before the course started. 86% did not have such self-confidence; only a small group of students (14%) were confident in programming before the course.

Question: 1 My perceived current level of programming knowledge is.

<table>
<thead>
<tr>
<th>Level</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19 (52.8%)</td>
</tr>
<tr>
<td>2</td>
<td>8 (22.2%)</td>
</tr>
<tr>
<td>3</td>
<td>9 (25%)</td>
</tr>
<tr>
<td>4</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>5</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>

Question 2: I took a programming course prior to this class (high school, computer camp, community college, distance course, etc.).

Question 3: I did not take a programming course before, but learned to program on my own by watching tutorials online, reading books, talking with others, etc.
Question 4: I had self-confidence in programming before I enrolled in the Client-Side Programming course.

4.2.4 Motivation
In this section of the survey, students were asked to answer questions related to their motivation to study programming.

Figure 9 indicates the number of students who have been encouraged or motivated by others before the course started, with 61% negative responses and 39% positive in the answers. Question 1 reflects the external environment that can influence the students’ motivation of study programming, while question 2 (figure 10) represents the students’ self-motivation of the study, the majority declared that they study programming because they know how useful it is, only 3% stated “no” in their choice.

Figure 11 presents the students’ motivation for choosing programming as their future career. 31% answered positively, and 69% did not have the desire when they did the survey. However, with question 4 (figure 12), 97% of the students believe that it is still good to know how to program for their future work even though it will not be a career.

The last question (question 5) in this section reflects the students’ motivation regarding the course itself. 89% of the students believe that they will succeed in the course when they did the survey, which was at the beginning of the course.

Question 1: Did someone give you encouragement or desire to study programming either by words or role modeling?
Figure 9: Students’ survey answer on question – Motivation 1

Question 2: I study programming because I know how useful it is.

Figure 10: Students’ survey answer on question – Motivation 2

Question 3: I want to develop a career as programmer in the future.

Figure 11: Students’ survey answer on question – Motivation 3

Question 4: It will be good to know programming for my future work even though I’m not planning to work as a programmer.
Question 5: I believe that I will succeed in the course.

4.2.5 Demographic Question
A demographic question is asked in the survey to support the understanding of the students’ academic background. More than half of the students participated in the survey are from Sweden.

Question: which country are you from?

- Bulgaria: 1
- China: 2
- Hungary: 1
- Chile: 1
- Netherlands: 1
- Latvia: 1
- Romania: 1
- Sweden: 34
- Thailand: 1
- Turkey: 1
- Kenya: 1

Figure 14: Students’ survey answer on demographic question
4.3 Interviews

4.3.1 Math Background

Low grades
The students with lower grades had different experiences with math in high school. When discussing their level of comfort in math and how they enjoyed it, the answers varied from “hate it” to “really enjoyed it”. However, many of the students commented about some parts of math being hard to understand or that they had only taken the required math course in high school. The vast majority of the students did not consider math knowledge as helpful in programming, commenting that there was not much math in the programming course, only basic math.

High grades
Among the students with high grades, the experience with math varied, where about half enjoyed math in high school, and the rest did not, with some commenting that they had taken advanced math in high school, but some only took the required math. Most of the students in the group did think math knowledge helps in a programming course referring to the logical thinking that is used both in math and programming. One student expressed oneself:

“How to think programming - definitely the math helps. I see people who are not good at programming and they are often not good at math, like the logical thinking.”

Math prerequisites of CS and Non-CS programme
To better understand the students’ math background, the latest entry points and required mathematical of New Media Design programme are investigated, in comparison with the other two computer science programmes at Jönköping University. Furthermore, the same comparisons were made with the CS and non-CS programmes in four other universities in Sweden. Programmes that were selected to present issue a bachelor’s degree and contain at least one programming course according to their education plans, which ensures an introductory programming course in each programme. In the tables below, the non-CS programmes are marked with a background color. Data in this section is retrieved from the universities’ websites and the Swedish Council for Higher Education website (Universitets-och Högskolerådet, 2020). Besides, the required math levels from 1a to 5 and from A to D, for example, level 2e stands for a lower level compared to 3b, level B requires less knowledge than level D (Skolverket, 2020).

<table>
<thead>
<tr>
<th>Programme Name</th>
<th>Entry Points (BI)</th>
<th>Required Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Media Design</td>
<td>18.91</td>
<td>2a, 2b, 2e or B</td>
</tr>
<tr>
<td>Embedded Systems</td>
<td>14.54</td>
<td>3c or D</td>
</tr>
<tr>
<td>Software Development and Mobile Platforms</td>
<td>16.30</td>
<td>3c or D</td>
</tr>
</tbody>
</table>

Table 4: math prerequisites of CS and Non-Cs programme at Jönköping University

<table>
<thead>
<tr>
<th>Programme Name</th>
<th>Entry Points (BI)</th>
<th>Required Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction Design</td>
<td>17.30</td>
<td>1a, 1b, 1c</td>
</tr>
<tr>
<td>Computer Science and Mobile IT</td>
<td>15.72</td>
<td>3c or D</td>
</tr>
</tbody>
</table>

Table 5: math prerequisites of CS and Non-Cs programme at Malmö University

<table>
<thead>
<tr>
<th>Programme Name</th>
<th>Entry Points (BI)</th>
<th>Required Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Experience Design</td>
<td>13.90</td>
<td>2a, 2b, 2e or B</td>
</tr>
<tr>
<td>Programming</td>
<td>15.05</td>
<td>2a, 2b, 2e or B</td>
</tr>
<tr>
<td>Computer Science</td>
<td>13.39</td>
<td>3b, 3c or C</td>
</tr>
<tr>
<td>Game Development</td>
<td>15.31</td>
<td>3c or D</td>
</tr>
</tbody>
</table>

Table 6: math prerequisites of CS and Non-Cs programme at Skövde University

<table>
<thead>
<tr>
<th>Programme Name</th>
<th>Entry Points (BI)</th>
<th>Required Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Design</td>
<td>*(all)</td>
<td>2a, 2b, 2e or B</td>
</tr>
<tr>
<td>IT Security and Software Testing</td>
<td>*</td>
<td>2a, 2b, 2c or B</td>
</tr>
</tbody>
</table>
Table 7: math prerequisites of CS and Non-Cs programme at Dalarna University

<table>
<thead>
<tr>
<th>Programme Name</th>
<th>Entry Points (BI)</th>
<th>Required Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Development</td>
<td>14.02</td>
<td>2a, 2b, 2c or 3b, 3c, or B, C</td>
</tr>
<tr>
<td>IT Design – System Design</td>
<td>12.77</td>
<td>3b, 3c or C</td>
</tr>
<tr>
<td>Computer Technology(Datateknik)</td>
<td>*</td>
<td>3c</td>
</tr>
<tr>
<td>Computer Science (Datavetenskap)</td>
<td>*</td>
<td>3c or D</td>
</tr>
</tbody>
</table>

Table 8: math prerequisites of CS and Non-Cs programme at Karlstads University

4.3.2 Previous Programming Experience

Low grades
The common theme for the students with low grades did not have any programming or coding experience. One student expresses not having any idea what HTML was. Two students from the group of low grades had the experience of C++.

“I really didn’t have any prior knowledge in programming. So yeah, that’s that. I started getting in touch with programming as soon as I started this course but prior to that, no."

“We were studying mainly C++, but our teacher was very bad, so he didn’t really know well, and then we didn’t really care as well, so I know basic things, but not that much, and at some point, I wasn’t interested in it.”

High grades
In contrast to the group with lower grades, the majority of the students with high grades had some previous experience with programming. The majority had experience with object-oriented programming languages and some even with JavaScript, which is the language of the Client-Side Programming course. Two students did not have previous programming experience, but one of them had experience with HTML and CSS. Most students with previous programming experience stated that they enjoyed the subject when they first studied it, and also expressed themselves in a more informative and enthusiastic way.

“In my free time I do a lot of [...] programming, so I do have some experience and probably that’s why I enjoyed [the course]”

“When I started high school in 8th grade we had C which is like a really low-level language, and that was like alongside of a basic introduction of computer programming in general, and grade 9 until grade 11 I believe, we had Java, so I studied Java like pretty intensively, and also like MySQL, databases, that was pretty fun and we get to do all of the projects. And then in 12th grade we had web development which was HTML, CSS and JavaScript. But at that time, I wasn’t really interested in JS. And then my last exam, kind of like my high school exam was in CSS and HTML.”

“I have never written a single line of code before I started [the programme].”

4.3.3 Comfort Level

Low grades
The answers from students with low grades showed a common theme in not feeling comfortable asking questions to the teacher during lectures. Most students in the group express that they would not ask or instead ask students in the class that had a higher understanding of the content.

“I would ask one of my friends that was a programmer. I wouldn’t ask the teacher because I kind of feel stupid sometimes”
“I usually try to figure it out myself, because I’m not very comfortable with asking people for help, especially teachers”

“I would ask my classmates that are good at programming”

When asking about their comfort level in labs - the same theme is evident with multiple students not being completely comfortable asking questions; however, some students also explain that they do feel comfortable during labs and that they would ask the tutors there with the focus being on who the person they asked were, for example, a friend or simply someone they felt comfortable with.

“Sometimes I think I am a little bit shy, thinking of ‘do they think I am stupid?’”

“I asked some of the lab assistants. Because one of my friends would always help me. I wouldn’t ask the teacher”

“Sometimes I am not very open to people, so it depends on the lab assistants themselves, who they are. So, if I’m more comfortable with them I would be willing to ask questions.”

“Sometimes [I did feel comfortable asking questions in labs], sometimes I just asked my friends that understand, and they show their own things and I try to figure it out”

The majority of the students expresses not having asked the teacher questions directly, for example through email or booking a meeting with him. However, some of them say that they would ask the teacher questions during the lab instead.

High grades
The group of students with higher grades showed a willingness to ask questions if there was something they didn’t understand. With the majority showing a positive attitude towards the lectures and some students expressing that they would directly ask the lecturer to explain the part that was unclear to the student. Some students say that they had specific methods for figuring out things after the lecture instead, like searching the web, watching tutorials online or read through the literature connected to the lecture.”

“I always go to the teacher and ask in class. I guess I wouldn’t ask in class if it was something more complex. I would go after the class and discuss with [the lecturer] without taking time from the lecture. But yeah, I feel comfortable asking questions in general.”

“I would ask the teacher to explain it more or ask a friend if they could explain it. I also use YouTube videos to get a better understanding of the topic of the class.”

“I think I tried to go back to the lecture [slides] afterwards, like I thought the literature was very good so I would read the book.”

When it comes to labs, the students with high grades show a higher willingness of asking a question—expressing a positive attitude towards the labs.

“Especially during the labs, we have extra students who can help, they are very helpful, most of the time it’s just something small you have to ask. Like you don’t see a dot or something written in your code and they help you see that. So, I think they are very good. And when I am on my own that’s Stack Overflow, it’s your saving net.”

“I actually think [the labs] were pretty good. The assistant was like students from last year, they always walked around and helped. So, I really felt comfortable asking them at any time with anything.”
Concerning asking the teacher questions directly, for example, through email or going to the teacher’s office, the group of students with high grades also showed a positive attitude. Most of the students had contacted the teacher personally, either through discord (a gaming communication tool), email, or going to his office. The students who said that they had not done those things commented the reason behind that being either that they did not feel the need to contact him or that they usually want to solve the problem themselves without help.

“I don’t think I would have had any problems with [contacting the teacher] but I didn’t do it that much. I usually think that I want to solve it myself if I can, so that’s probably why. I usually ask when it has to do with the instruction of the assignment, but not how to solve it.”

“No, I never felt the need [to contact the lecturer] because I understood quite a lot, so I never had problems with any of the labs.”

4.3.4 Motivation

Low grades

When asking the students with low grades about their motivation to learn to program – a common line of argument was presented through the interviews. All the students expressed why they chose to study programming as connected to the digital world we live in today. A common way to describe this was by using words/expressions like “useful”, “good to know”, “relevant to our lives” and “more opportunities”.

“I think it is important to know something like this, but for me I don’t feel like I would have a job in programming, but I think maybe it would be useful at some point.”

“I feel like if I’m going to do anything that has to do with computers, I have to gain a deeper understanding. I think it is good to know but I wouldn’t use it necessarily. My dream isn’t to be a programmer.”

“I studied [programming] to have more opportunities to get a work. For me, I don’t want to work directly with programming. I want to work with design and have this experience of programming”

“I didn’t really know a lot of programming before. But I heard from my older sister that it is quite good to know programming at least. So, I thought why not. It’s pretty hard, but it’s actually good to know it for future reference.”

The students with low grades have in common that they do not see a future career within programming. Most of the students are interested in design (the other main subject of the programme in which they are enrolled). Some students make comments about why programming is not a path they will take because of the difficulty of programming. However, many students express having chosen to study the specific programme due to its combination of design and programming.

“I would like to be a graphic designer, or something related to that”

“I think I would preferably work with design, and less things programming.”

“I wouldn’t want to work with programming in the future. I feel like there are better people made for that. [...] I am more of a visual kind of person.

High grades

A word used by all students who received a high grade when discussing motivation to learn programming was “fun”. All students expressed that they thought programming is fun and that they enjoy doing it and that that is what gives them the motivation to learn.

“I think it’s fun. I actually enjoy programming more than a lot of people do it. This isn’t like the top-tier programming, you can go so much more extreme, and I like that it is kind of chill”
“I think it’s pretty fun. In general, just getting an idea and being able to realize it and to do it. It is really satisfying in my opinion, like taking over a problem and coming up with a solution and thinking really logically. Things get stuck but then when you solve it is like really rewarding. Yeah, I just enjoyed the logic and fun parts behind it, the challenge behind it.”

Regarding their future career paths, the students who received high grades all saw programming as an option for a future career. Many students were interested in combining design and programming in their future work.

“I could definitely think something with programming, sure. And since I’m going in this course, I think graphic design could be fun. But programming I think, I’m leaning more towards that.”

“I haven’t done a definite decision but definitely within some kind of either like towards graphic design, UX, or frontend development. It sounds like a lot of fun combining design and programming together.”

“I would say that my primary motivation now for learning programming [is to] get a job at a big company in Sweden with programming. [...] I would definitely say that my career prospect is my main motivation when it comes to logical thinking and solving challenges.”

The combination of design and programming in the programme in which the students are enrolled is mentioned as a positive attribution to the choice of school for most of the students in the group.

4.3.5 Attribution to Success

Low grades

The students with lower grades on the exam explain their result in different ways. With the interview question being “What do you think are the reasons you got that grade” after having answered which grade they received, some students focused on the positives - they actually passed - and attributing that success to having studied and watched YouTube videos on the subject which is attribution to effort. Most students however focused on the negatives, why they didn’t receive a higher grade. One student says that s/he didn’t really pay attention and couldn’t concentrate when s/he saw the exam suggesting attribution to ability. One student expressed that the reason for not getting a higher grade was due to not being able to solve complicated questions, which is considered as attribution to the difficulty of a task. Others blame not having put in enough effort in the course, meaning that they attributed their lack of success to effort. One student said they couldn’t remember a lot of things on the exam and another state that they tried to solve some tasks but gave up instead.

“I always had someone helping me, and during the exam I was by myself. I didn’t put as much effort in the [Client-Side Programming] course as I put in other courses.”

“I couldn’t do like part of programming like loops or something like that. And like the complicated questions to coding”

“I focused more on the theory questions than on the practical ones. I sort of tried to program some of [the programming assignments on the exam], but in the end I gave up instead.”

Multiple students expressed an attribution to the difficulty of a task when discussing which grade they were aiming for in the course. They said that before the course started, they had been aiming for a higher grade, but that the difficulty of the programming assignments made them lower their expectations on themselves.

“At the beginning I tried to get [the highest grade] and then the labs started to get harder and harder, so I started lowering my aiming.”
“At first I was aiming for [the highest grade] and then the course became hard and then I was aiming for [passing the course].”

All of the students in the group with lower grades on the exam regard themselves as designers instead of programmer, except for one who identifies as both. Many students comment about certain characteristics that make them more of a designer or less of a programmer, attributing their lack of success in programming to ability.

“I definitely find myself more artistic. I really like making things with my hands. Programming involves more brain skills, while art is more visual skills, so that’s why I like [design] more.”

“I don’t consider myself as too much of a logical thinker”

“I kind of like lack the skills to be able to define myself as a programmer.”

High grades
In a discussion about why the interviewed student received a high grade on the exam - most of the students attributed their success to ability or effort. One student pointed out that they liked “doing” and that they were able to get into the right mindset needed for programming and was able to think logically. This shows that the student attributed their success to ability. Many students commented that they studied a lot, “tried hard” or read a lot of literature on the subject. One student thought that his/her success was a result of having previous programming experience.

“I definitely think that [getting a high grade] was because of my previous programming knowledge, like the logic part of it. I believe if you studied programming before an you understood the logic, in any language, it doesn’t matter which language you study after. [...] And I studied as well, like I read a lot online to figure it out”

“The reason I got [the highest grade] on the exam may have been that I actually know [the programming] “by heart” instead of only being able to solve on programming task at a time. Instead of like only studying for the exam, actually knowing it.”

All of the students but one that received a high grade on the exam said that they aimed for the highest grade before the course started with one student attributing success to ability and effort:

“I just like to challenge myself when it comes to programming, and I know I can do this. I knew if I sat down and studied and tried to solve coding I can get [the highest grade].”

All students with high grades regard themselves as programmers before designers, except for one that identifies themselves as both. The students with high grades show attribution to ability when discussing why they regard themselves more as programmers. They argue to be better at programming than design, and being more comfortable with programming, as well as programming coming easily to them.
4.3.6 Demographic Statistic of the Interviewees

![Bar Chart]

Figure 14: Demographic Statistic of the Interviewees

4.3.7 Observations of the language and wording used by interviewees

All the interviews were conducted in English, while all the interviewees do not use English as their mother language. Observations were made during the interviews on the students’ talking style and language skills, which have shown patterns in the different grade groups. In general, students in the high-grade group possess relatively better skills in English, and most of them were relatively more assertive and specific when answering the questions. Students in the low-grade group usually had short answers and were relatively vaguer and more reticent when expressing themselves, especially when it came to the topic of programming.

Both groups have interviewees that were talkative and cordial but have different preferences when choosing their words. Most of the high-grade interviewees showed confidence in their wordings, such as “I definitely…”, “I never had problems with any…”, “I was well aware that…”, “It came so easy to me…”, etc. In the low-grade group, many of the interviewees showed nonproficiency of their English skills, also talked with less assertiveness and confidence. For instance, they used phrases like “I wouldn’t really…”, “I survived…”, “we didn’t really care…”, “maybe it would be useful at some points”, “I don’t know…”, etc.
5 Analysis

5.1 How does math background contribute to success in introductory programming in a non-CS major?

The result of the survey reflects that more than half of the students perceived themselves with a weaker math background; they enjoyed the subject but did not find it easy as a quick learner. Connecting with the required math level and entry points of New Media Design program, it could not be proved that the students who found math easy possess a strong math background just based on the data collected in this study. The required math 2a, 2b, 2c could be the general level of math for students who studied in a Swedish high school but not international students. What’s more, math 2a, 2b and 2c are not the high levels of math in the Swedish education system.

However, the NMD programme has in fact the highest entry points 18. 91 (out of 22.5 in total) among all the CS and Non-CS programs in the five universities mentioned in this study, so even though it could not be proved that the NMD students have solid math background, their learning skills are proved to be relatively stronger and have the ability to study the subjects taught in the program.

Investigating students’ perception of math and how comfortable they felt with math in high school during the interviews did not give us any clear result of how math background would have an impact on the success in an introductory programming course. The comfort level/enjoyment in math courses in high school varied in both groups, with some students expressing a negative view towards math and others the opposite. The only noticeable difference between the two groups - low grades and high grades - would be their attitude towards how math helps with programming skills. Most of the students with low grades did not think math knowledge helps, saying that only basic math was required in the introductory course. Most of the students with high grades however connected math and programming with the logical thinking that must be present in both areas, arguing that through that perspective - math knowledge is helpful in a programming course.

Furthermore, the different programme prerequisites of math reflect that the Non-CS students possess a weaker math background, in general, compare to students who study in CS program, which makes this factor highly relevant to the research topic even though many researchers have declared math background as one of the most influential factors in programming learning (Campbell and McCabe, 1984; Wilson, 2002; Bergin and Reilly, 2005). It shows in the data that the CS programme requires the students to have studied at least math 3b, 3c or C, D as the prerequisite levels, which ensures the students deeper knowledge in the subject. This could relate to another finding from the interviews, that many of the interviewed students show a tendency to enjoy social science subjects more than subjects that involve logic, data and evidence. Some reveal characteristics such as “imaginary”, “creative” instead of being accurate or meticulous in logic. Even fewer declared that they enjoy both logic and creativity, but there is no clear pattern on those students who perform successfully in both design and programming courses in connection with math background. An assumption could be made here that most of the interviewed students have the capability to learn math well but are more interested in non-natural science subjects. This will be discussed and integrated into the analysis of motivation factor later.

5.2 How does previous programming experience contribute to success in introductory programming in a non-CS major?

The survey result indicates that most participants do not have any experience or knowledge prior to the introductory programming course and lack confidence in programming. It could be seen obvious that more than 2/3 of the students belongs to the non-experienced group.

When comparing the interview answers from students who succeeded in the course and students who got lower grades, previous programming experience indeed contributes to the success in introductory programming. Among the interviewees who got low grades in the
course, five out of seven have no prior experience in programming. This looks different in the high-grade group, where only one student had no experience in programming before the course started.

Nevertheless, the interviews have revealed a pattern that experience is not the absolute factor to learn to program well. One participant with three years of high school education in computer science who studied object-oriented language got a low grade in the Client-Side Programming course because of other factors. While one student with absolutely no experience in programming before the course found the course easy and succeeded with the highest grade.

In general, previous programming experience did show positive influence on the students’ performance in an introductory programming course, but since the students gained their knowledge through various schools, teachers and classes in their home countries, also had different level of interests in the subject itself which also affected their level of knowledge, it is hard to draw a conclusion of whether the general programming experience has a correlation to the success in an introductory programming course, but could be more specific to different type of experiences. For example, regarding the programming experience from high school courses, unlike what Ayalew et al. (2018), Bergin and Reilly (2005) and Ventura (2005) have found in their research, the data in this study shows that the students’ programming experience in high school does affect their performance in introductory programming course at university level. Those who had programming courses in high school and enjoyed them also did well in the introductory programming course at university compare to the one who just survived the programming courses in high school.

Wilson (2002)’s finding which declares that previous experience is not a significant prediction of success on the whole course but influence more on the midterm of the course, cannot be proved in this study. One of the reasons is that there is no midterm exam in the Client-Side Programming course, so there is no data to compare with the final exam. However, students have described their thoughts on the grades before and after the course. Many students with no previous programming experience stated that they were aiming for the highest grade at the beginning however lowered the standard gradually during the course, and most of them indeed ended up with the lower grades. The interviewees who got higher grades, also with the most experiences in programming, declared that they wanted the highest grade and had confidence since the beginning of the course, things did not change much for them before and after. It could be seen that in this study, the lack of previous programming experience is not an obstacle for the students at an earlier stage of the course, however, served as an important support to the later stages of the course when the content became more complex.

The study did not discover any differences between students’ demographical backgrounds and the correlation to previous programming experience as a success factor. Sharma and Shen (2018) found that previous programming experience is more influential to the Australian students as a success factor than the Indian students. However, in this study, five Swedish students were interviewed; the rest are from Europe, East Asia, Latin America and Oceania. There was no clear pattern in the interviews that one region of the students showed a significant difference than another. It could be only noticed that students from Sweden and Europe have more experiences in programming than the rest of this study.

Besides, according to the interviewees’ answers, those who succeeded with the support of previous experience had shown a more solid understanding of programming and genuine interests in the subject, while the one who failed with experience did not like the subject before and just had to go through the course. The student also declared in other sections of the interview that it is more pleasant now with graphic design courses compared to the student’s high school time when there were only programming courses.

5.3 How does comfort level contribute to success in introductory programming in a non-CS major?

The interview data showed signs of students with low grades having a lower level of comfort in the course, compared to the students with high grades. Many of the students with low grades expressed not being comfortable asking questions during lectures because of the fear of being looked upon as stupid. In the group of students with high grades, there was contrasting results.
of the students being more comfortable with asking questions in general, or as two students expressed – not having the need to ask questions directly to the lecturer because of the students knowing how to solve the assignments independently. Furthermore, the students with higher grades seemed more comfortable in knowing how to tackle the problem of not understanding something by themselves, with methods like searching the web, watching tutorials online or read books on the topic.

To answer the research question “Does comfort level contribute to success in introductory programming in a non-CS major?”, the data from the interviews suggests that comfort level could be seen as a success factor. The students who showed a higher level of comfort did indeed perform better than the students with a lower level of comfort, and the results from the interviews are supported by the study by Wilson (2002), in which comfort level was found to be the most significant success factor out of the 12 factors studied.

However, a few students showed that having high comfort level does not ensure them high-grade in the course, the general difference between them and the students with a high comfort level and high grades is that they expressed tendencies to be dependent on others’ help. Those students stated that they liked to ask friends and the lab assistants often or even get to see the finished assignments from others, while high-grade students tend to try their best on independent problem-solving. A few low-grade students showed that they could feel comfortable in the course and were not afraid of asking questions or being judged by others naturally, even though it is not a subject they are good at or interested in.

It is noticed that these students with high comfort levels and low grades presented more social skills during the interviews and tended to talk in a relatively optimistic style in their language and gestures compared to those with low comfort levels and low grades. As for the high-grade students, a common pattern is that they tended to talk and think in a rational way during the interview and liked to define themselves as a rational person or understanding logic well. Table 9 presents a summary of the students’ revealed cognition style patterns during the interviews. Although their cognition styles could not be judged by their performance in a single interview, there are still some signs that cognition style might affect their comfort level in a programming course, even contribute to the success in the course. According to Akinola et al. (2011)’s study, student-programmers with MBTI cognitive style ISTJ tend to have better programming skills than others, some attributes of the ISTJ cognitive style (e.g. realistic, analytical, objective) could also be found in the same group of interviewees in this study. For instance, students with higher grades usually self-define as a rational person, while the lower grade group tend to declare themselves as artistic or creative.

When using the comfort level as one of the success factors, combining students’ cognition style could be considered to improve the accuracy of the assessment. More recommendations will be provided in the next chapter.

<table>
<thead>
<tr>
<th>Grade Level</th>
<th>High Comfort Level</th>
<th>Low Comfort Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Grades</td>
<td>Open, talkative and social / Not afraid of asking / Do not like logic or math / Declaring themselves as artistic or creative</td>
<td>Shy to ask questions / Of few words / Do not like logic or math / Declaring themselves as artistic or creative</td>
</tr>
<tr>
<td>Higher Grades</td>
<td>Outgoing and open / Do dare to ask / Self-define as a rational person / Passionate about programming</td>
<td>Quiet but dedicated / Seem to have strong self-discipline / Passionate about programming / Self-define as a rational person</td>
</tr>
</tbody>
</table>

Table 9: summary of the students’ revealed personalities in the interviews.
5.4 How does motivation contribute to success in introductory programming in a non-CS major?

The survey results indicate that almost all students in the programme have some motivation to learn programming, with 97% stating that they study programming because they know it is useful. Also, 97% of the respondents think that it will be helpful to know programming in their future work even though they would not end up working as programmers. Looking closer on career driven motivation – the survey results show that only 31% of the students in the class answered yes to the question of whether they want to develop a career in programming in the future, which could be an indicator to as of why only 24% of the class received higher grades. This, however, is just speculation as the survey was anonymous; we don’t know from the survey results that the students who want to develop a career in programming were the ones who received high grades.

Although all the students in the interviews – in both groups with low and high grades, expressed having chosen to study the programme of New Media design (graphic design and web development) with the motivation that the mix of design and programming is good to have, there was a clear distinction between the groups in their plans for the future. The argument for studying programming for the students in the low grades was in general only because it is “good to know”, whereas the students with high grades all saw a future in programming. Only one of the seven students in the group with low grades considered oneself as a programmer, whereas all the students in the group with high grades considered themselves as programmers. Also, all the interviewees in the group with high grades referred to programming as being “fun” and that that was a motivation of studying it.

The study by Zainal et al. (2011) showed results confirming motivation to be a success factor in an introductory programming course, as students who showed an interest in programming performed better. However, Sharma and Shen’s (2018) study was made between two different universities in two different countries, where only results from one university showed results of motivation is a significant success factor. This is especially interesting in the context of the study since the students being interviewed are from countries all over the world, and that was not something that was considered when comparing the groups of low and high grades. The study by Sharma and Shen (2018) suggest that the outcome could have been different had culture been taken into account.

In summary, motivation was found to contribute to success in the introductory programming course, with the major differences between the groups being that the student who performed better expressed that motivation for doing programming was that it is “fun” and saw a future in working with programming.

5.5 How does attribution to success contribute to success in introductory programming in a non-CS major?

The result from the interviews about attribution to success did not show a very clear distinction between students with low and high grades. There was no clear pattern among the students in the same group, either. In the group with low grades, the students attributed their success to three different causes. Some students attribute their success/lack of success to effort, some to ability – not being able to concentrate, or not being able to do certain parts of programming, for example. Only one student attributes his/her lack of success to the difficulty of the task.

Wilson (2002) found in her study that attribution to success was one of the significant success factors in introductory programming, but that it was something that influenced the performance negatively – namely attribution to the difficulty of task and attribution to luck. Students who attributed their success to these two factors were more likely to perform on a lower level. Our study does to some extent support this, as there were no students with high grades that attributed their success to these two attributions but there was also only one student in the group with low grades who explicitly said mentioned that the questions in the exam were “complicated”. However, some students with low grades said that they at the beginning of the course had aimed for the highest grade and that when the course became harder, they had lowered the aiming to just passing the course. This also suggests attribution to the difficulty of the task, as the students did not choose to keep aiming for the highest grade when the subject became harder to understand.
The part that seemed to be most significant to attribution to success through the interviews was attribution to ability. The students with low grades all identified as designers over programmers (except for one who said both) and discussed not having the characteristics that according to the students – a programmer should have, for example, logical thinking. The students expressed being more “artistic” suggesting that being artistic was something that did not match well with programming. Multiple students with high grades attributed their success to ability as well, with all of them identifying as programmers more than designers. One of them also commented that the reason for achieving a high grade on the exam was due to knowing programming by heart and another that you had to have a specific mindset.

In conclusion, the results from the interviews on attribution to success were a little bit scattered, and attribution to success is not considered as an absolute factor of success according to the results in this study. However, what was visible was that students with lower grades were more likely to attribute their lack of success to the difficulty of the task than students with high grades, and both groups would attribute their success/lack of success to ability.
6 Discussion and conclusion

This chapter summarizes the results of the study. Further, it describes the implications and limitations of the study. It also describes the conclusions and recommendations of the study. Suggestions for further research are given at the end of the chapter.

6.1 Findings

How does math background contribute to success in introductory programming in a non-CS major?
Math background as a success factor for introductory programming in a non-CS major could not be proved by the data collected in this study. Initially, when designing the research, the differences between math educations in different countries were not considered thoroughly. It would be more reasonable and credible to compare the math level with students from the same education system, for instance setting a question in the survey for the Swedish students regarding the highest level of math course they took in high school. Besides, there is no clear pattern to be found in the interviews either. Students from both the lower-grade group and the higher-grade group have mixed opinions that could not support a clear answer to the research question.

How does previous programming experience contribute to success in introductory programming in a non-CS major?
Previous programming experience does contribute to the success in introductory programming in a non-CS major, but not as the most dominant factor. Students without any experience could still succeed in the course, but most higher-grade students do have experience in programming before the course started. What is more, the data in this study indicate that introductory level of programming in a non-CS major does not require long-term programming experience to be able to perform well in the subject.

How does comfort level contribute to success in introductory programming in a non-CS major?
Comfort level could be a success factor in introductory programming in a non-CS major. The study showed that high comfort level is also highly relevant for a student’s success in the introductory programming course. However, when analyzing the factor, possibilities are found that students’ cognition style might be another factor hidden behind which affects not only their comfort level but also their performance in the course in general.

How does motivation contribute to success in introductory programming in a non-CS major?
Motivation does contribute to the success in introductory programming in a non-CS major. Students who performed better in the course showed a strong tendency to wanting to have a career within programming in the future. While the lower-grade students have more statements of programming being useful but not required for their future career, a clear distinction could be found in the students’ reactions that motivation is a success factor with high certainty.

How does attribution to success contribute to success in introductory programming in a non-CS major?
A solid proof of attribution to success being a success factor could not be supported indisputably by this study. A common pattern was found that students tend to summarize the attribution to ability as their attribution to success. Logical thinking is one of the most significant abilities they described while many who got lower grades in the course defined themselves as more creative or artistic, which contracts to the abilities of being logical and rational.

6.2 Implications

Many universities in Sweden have non-CS majors on both bachelor and master level that include introductory programming and could benefit from an understanding of the students who are enrolled in these programs and what makes the students successful in the course. The results of this study imply that some success factors can be considered when designing an introductory programming course for a non-CS major.
Find ways to motivate non-CS students in introductory programming in combination with their own interests
Motivation is a highly significant factor in introductory programming. If the introductory programming course includes parts about how programming will be used in their future work, students may feel more motivated to learn and therefore perform better in the course. An example could be to use external lecturers who explain how the industry works. Maybe the lecturers could be alumni from the programme that can explain in what way they use programming today and their recommendations for the current students.

The results of the study suggested that most students with lower grades did not plan to work as programmers in the future, and with 76% in the class having lower grades, an assumption could be made that most students do not have a very strong motivation to learn to program. This could help the teacher structuring the course with how to present the content. For example, many of the students with lower grades in the interviews identified as designers and many said that they had “artistic” characteristics rather than logic thinking. Knowing this, the teacher could find appropriate teaching methods for this group of people. For example, the artistic students may prefer a visual approach to learn programming concepts, instead of presenting them in a traditional computer science way.

Customize course content for non-CS majors with the help of MBTI
The lecturer can design various assignments to the different groups of students after investigating in their purposes of studying the course. Students who regard themselves as non-programmers could select assignments that are more relevant to other subjects in the major. For instance, in a major that combines design and programming, students who declare themselves as designers could learn the programming language by coding a piece of design. An introductory programming course in a non-CS major could be customized in a more adaptive and flexible way that better consider the students as non-CS majors. An idea should be given to them that the course is not the same level of an introductory course in computer science major, and how they could adapt the programming knowledge to their own fields. Furthermore, the MBTI model could be used as a reference for those students who do not know which path they should take in the course. Someone that has a cognition style linked to higher possibilities to enjoy programming could be suggested to try out assignments in higher levels at the beginning of the course.

Offer bridging courses or lectures before introductory programming
The school could also consider offering the students bridging courses before an introductory programming course start. For example, bridging courses in math could help students get trained in logical thinking. What is more, lectures could be provided to help non-experienced students understand the terms that are frequently used in programming so they could follow the lectures with fewer efforts at the beginning of the course. Tools and trustworthy study resources could be introduced to students at the earlier stages of a course (e.g. Stack Overflow, MDN, Visual Studio Code, etc.) to present them the problem-solving process supported by the different tools and resources so that they could be more independent when to try to solve programming problems.

Offer various forms of mentoring to increase students’ comfort level
Comfort level is also a success factor that should be considered when designing an introductory course. The results of the study showed differences between students who felt comfortable to ask questions and those who did not. The students with lower grades expressed that they felt more comfortable with people they knew and that they sometimes felt stupid when asking questions. This suggests that it is important that the teacher offers various forms for students to ask questions. In the case of student assistants attending labs, these assistants should be informed about how to act in front of the students who need help. What to say and what not to say and consider body language to make sure the students feel comfortable.

Lastly, the factor that is significant to success in an introductory programming course is previous programming experience. This is something that could be harder to find a direct implementation of, as previous programming experience is not required to enroll. Maybe it could be used to create study groups for the students at the beginning of the course, where students can learn from each other.
Ideally, further research should be done on the specific success factors that have shown to be significant, to examine specific ways how to take advantage of the factors when designing an introductory programming course.

6.3 Limitations

The scope of this thesis has resulted in some limitations. The most obvious is the small sample of data being analyzed. The data is only collected from students at one university and in one program. Ideally, at least students in similar programs from different universities in Sweden would be part of the study to be able to generalize the results better. Another thing that could be limiting is that the students in the study had different nationalities and therefore came from different previous education systems throughout the world. This may be something that influences the students’ perception and understanding of programming and therefore, also their performance, as the course was part of the Swedish higher education system.

The small sample of data also resulted in a grouping of the students based on low grades and high grades, where a grouping of each individual grade could have resulted in a more accurate result on success factors. The grouping of low grades and high grades is less specific.

Another part that was found to be limiting was the method for determining the contribution of success in math background and previous programming experience. Because of the students being from different parts of the world with different backgrounds, the two success factors previous programming experience and math background were hard to analyze. To better research these, the research could either let the participants take a standardized test before the introductory programming course, that makes determining different levels of math and programming knowledge easier, or focus on one nationality where for example the high school level math courses are comparable between the study’s participants.

6.4 Conclusion

The study has examined the five success factors for introductory programming courses in a non-CS major, with some results supporting the previous research and some contradict to them. Math background could not be proved as a success factor in this study due to the lack of a standardized assessment of the students’ math levels. Previous programming experience could be regarded as a success factor but not as dominant as the success factor motivation, which has shown clear patterns in the data. Comfort level could be seen as one of the success factors as well while most of the data in this study support to this conclusion. Lastly, attribution to success as a success factor could not be supported by this study; the qualitative data showed variety which makes it hard to draw a conclusion directly. To sum up, three success factors were proved to be effective in introductory programming courses in a non-CS major. Among them, motivation as a success factor has shown the most influence and is recommended to be seen as the most significant.

6.5 Further research

Throughout the analysis of the thesis, a few ideas for future research emerged. For one, many of the participants in the interviews had different nationalities as they are enrolled in an international program. With Sharma and Shen’s (2018) study in mind, where they compared success factors in introductory programming in an Indian university versus an Australian university, it would be interesting do recreate a similar study as this, but grouping students by multiple nationalities and seeing if any of the success factors play different roles depending on the students’ cultures.

The non-CS students’ previous programming experience could be evaluated by a standardized assessment; the study result would achieve higher validity if a quantitative study has been done. A suggestion would be to design an assessment of previous programming skills based on the average knowledge level of introductory programming course in a non-CS major.

Besides, MBTI tests could be considered to use in the investigation of non-CS students’ common cognitive styles. Are there any ways to help the students to better position themselves and find their motivation to study programming? Further research could examine the connection
between the MBTI cognitive styles and the success in an introductory programming course for non-CS majors. Practical suggestions could be provided to the students before the course starts based on their individual cognitive styles and interest.

Future research could also be done on different methods of implementing the results of the study. Is there a way to improve the students' performance in an introductory programming course based on the success factors that have been found significant?
References


Appendices

Appendix 1 – Interview guide

Appendix 2 – Survey questions
**Interview Guide**

**Math background**
(1) Were you comfortable in and/or enjoyed math courses in high school? (Age range as well to clarify)
(2) Do you think your prior math knowledge helped you in the course?

**Previous programming experience**
(3) Describe your experience with programming before the course started.

**Comfort level**
(4) What would you do if there was something you didn’t understand during a lecture?
(5) Did you feel comfortable asking questions in labs? Why?
(6) How comfortable were you asking questions directly to the lecturer outside of class? (e.g. through email or going to the office)

**Motivation**
(7) Describe your motivation to learn programming?
(8) What would you like to work with in the future?
(9) What made you decide to study New Media Design?

**Attribution of success**
(10) What qualities or skills do you think are important to have to learn programming well?
(11) Did you aim for a specific grade?
(12) What grade did you receive on the final exam?
(13) Are you happy about the grade you received?
(14) What do you think are the reasons that you got that grade?
(15) What is your perceived understanding of programming compared to other students in the class?

**Programming in a non-CS major**
(16) Do you regard yourself as a designer or a programmer?
(17) Designer: Why do you think you are more of a designer instead of programmer, what attributes or characteristics on you makes you feel like that?
(18) What are your thoughts about mixing design and programming in a university degree?

**Others**
(19) How did you enjoy the course?
(20) Describe your self confidence in programming before and after the course?
Appendix 2

Contact information

- Would you be willing to participate in a follow-up study at the end of this course? Please provide an email address where we can reach you.

Demographic & Math Background

- How old are you?
- Which country are you from?
- I enjoyed studying math in secondary school. (Yes/No)
- I find math easy and I’m a quick math learner. (Yes/No)

Previous Programming Experience

- My perceived current level of programming knowledge is [1/2/3/4/5]
- I took a programming course prior to this class (high school, computer camp, community college, distance course etc.) (Yes/No)
- I did not take a programming course before, but learned to program on my own by watching tutorials online, reading books, talking with others, etc. (Yes/No)
- I had self-confidence in programming before I enrolled in the Client-Side Programming course. (Yes/No)

Motivation to study programming

- Did someone give you encouragement or desire to study programming either by words or role modelling? (Yes/No)
- I study programming because I know how useful it is. (Yes/No)
- I want to develop a career as a programmer in the future. (Yes/No)
- It will be good to know programming for my future work even though I’m not planning to work as a programmer. (Yes/No)
- I believe that I will succeed in this course. (Yes/No)