Exploring and Expanding Students' Success in Software Testing

Deepti Mishra
Department of Computer Science, Norwegian University of Science and Technology,
Gjøvik, Norway
Sofiya Ostrovska
Department of Mathematics, Atilim University, Ankara, Turkey
Tuna Hacaloglu
Department of Information Systems Engineering, Atilim University, Ankara, Turkey

Abstract

Purpose: Testing is one of the indispensable activities in software development and is being adopted as an independent course by software engineering departments at universities worldwide. In this paper, an investigation of the performance of learners about testing is carried out, given the tendencies in the industry and motivation caused by the unavailability of similar studies in software testing field.

Design/methodology/approach: This study is based on the data collected over three years (between 2012 and 2014) from students taking the Software Testing course. The course is included in the second year of undergraduate curriculum for the Bachelor of Engineering (Software Engineering).

Findings: It has been observed that, from the performance perspective, automated testing outperforms structural and functional testing techniques, and that a strong correlation exists among these three approaches. Moreover, a strong programming background does help towards further success in structural and automated testing, but has no effect on functional testing. The results of different teaching styles within the course are also presented together with an analysis exploring the relationship between students' gender and success in the software testing course, revealing that there is no difference in terms of performance between male and female students in the course. Moreover, it is advisable to introduce teaching concepts one at a time because students find it difficult to grasp the ideas otherwise.

Research Limitations: These findings are based on the analysis conducted using three years of data collected while teaching a course in testing. Obviously, there are some limitations to this study. For example, student's strength in programming is calculated using the score of C programming courses taken in previous year/semester. Such scores may not reflect their current level of programming knowledge. Furthermore, attempt was made to ensure that the exercises given for different testing techniques have similar difficulty level to guarantee that the difference in success between these testing techniques is due to the inherent complexity of the technique itself and not because of different exercises. Still, there is small probability that a certain degree of change in success may be due to the difference in the difficulty levels of the exercises. As such, it is obviously premature to consider the present results as final since there is a lack of similar type of studies, with which the authors can compare their results. Therefore, more work needs to be done in different settings to draw sound conclusions in this respect.

Orginality/value: Although there are few studies (see, for example, Chan et al., 2005; Garousi and Zhi, 2013; Ng et al., 2004) exploring the preference of testers over distinct software testing techniques in the industry, there appears to be no paper comparing the preferences and performances of learners in terms of different testing techniques.

Keywords: *Software Testing, Functional Testing, Structural Testing, Automated Testing, Software Engineering Education*

1. INTRODUCTION

With the establishment of Software Engineering (SE) departments in various universities all over the world in the last decade, the need for having software testing courses has also emerged. Testing is considered as an important skill for SE graduates, a fact that the academic community has come to realize. Nevertheless, only a handful of papers have been published describing the insights and experiences of teaching this course effectively. On the other hand, there has also been increasing expectation by the software industry to instill this important skill into SE graduates as the economic impact of inadequate testing is still a major concern within the industry.
In the software sector, testers mostly prefer functional or specification-based testing over structural testing (Chan et al., 2005; Garousi and Zhi, 2013; Ng et al., 2004) because the latter is considered more cumbersome and time-consuming and can only be used at the unit-testing level; whereas functional testing is not only used during unit testing, but it can also be employed during integration, system, and user-acceptance testing. Additionally, these different techniques require slightly different skills, such as structural testing which requires delving into further details, while functional testing is more logical. Although it has been well established through various surveys and studies that functional testing is the most commonly used method for creating test cases by testers working in the software development industry, one may pose the question whether the same is true for learners as well, and whether they find functional testing easier. If so, should it be reflected in their performance evaluation. A more crucial question could be: whether their performances are better in functional testing than structural testing.
It has been reported that testing requires specific skills, including critical thinking, analytical abilities, and investigative skills. Strong programming background may have a positive effect on testing. Especially in structural testing, testers need to understand the code to create test cases. Similarly, automated testing requires a good command of programming. On the other hand, test cases are created using requirement specification in functional testing. Even though there is no need to know programming, still in order to derive the test cases consisting of inputs and expected outputs, there is a need to mentally visualize how the program satisfying given specifications will function. This process of understanding the basic logic is also an important first step of programming. Therefore, we need to investigate whether programming knowledge is also a beneficial factor to be a successful tester. Are better programmers better testers as well? Do higher programming skills result in higher rates of success in testing? More specifically, do higher programming skills help in conducting testing in general, or is this more related to success in a specific testing technique, e.g. functional, structural, or automated testing.
Later, the role of gender in the individuals' success is explored – an issue which appears to be still inconsistent. According to common gender stereotypes, women are assumed to have lower computer skills than men (Fleischmann et al., 2016). Despite intensive research going in this direction, there is no generally adopted view on the problem, with a variety of studies showing contradictory results. Therefore, it will be interesting to know whether male and female students achieve similar success rates during software testing. Further, are there any significant differences in their performance in a specific testing technique, e.g. functional, structural, and automated testing?
As stated before, universities have come to recognize the need for such a course. Yet, there still exists a gap in understanding as to how to teach this course effectively. While being introduced at different levels in undergraduate and graduate studies and in different departments in universities

around the world, the level of rigour/difficulty while teaching, the chosen programming language, and the testing tools depend on the level and orientation of students. There have been some studies (Chan et al., 2005; Edwards, 2004; Elbaum et al., 2007; Garousi, 2010; Mao, 2008; Smith et al., 2012; Timoney et al., 2008) reporting the experiences from software testing courses in universities, but the same teaching strategy may not work in every situation because of the reasons mentioned above. As such, there is a need to investigate the effect of shifting teaching strategies on the students' success. Despite a few studies (Chan et al., 2005; Garousi and Zhi, 2013; Ng et al., 2004) exploring the preference of testers over distinct software testing techniques in the software industry, there appears to be no paper comparing the preference and performance of learners in terms of different testing techniques. With these motivations, the authors intend to measure the performance of software engineering students by looking into the correlations among different software testing techniques, and possible implications.

2. LITERATURE REVIEW

There is lack of proper education in software testing in university software engineering programs (Chan et al., 2005). Software development organizations rely mainly on in-service software testing training by providing either internal training courses (61%), external training courses offered by commercial organizations (65%), or self study by staff (39%) (Chan et al., 2005).

2.1 Testing education

Cowling (2012) evaluated the courses relating to software testing in the Software Engineering volume of Computing Curriculum 2001, against a theoretical model that has been developed from a well-established program in software engineering. This is done with the perspective of how well the courses support the progressive development of both students' knowledge of software testing and their ability to test software systems. Barbosa et al. (2008) suggested that teaching of software testing should begin as early as possible so that an adequate culture of testing can be created. They proposed PROTEST – a Web-based environment for the submission and automatic evaluation of practical programming assignments based on testing activities. Elbaum et al. (2007) developed a web-based tutorial named 'Bug Hunt' to promote early integration of software testing concepts into the computer science curriculum in a manner that engages students, while making it amenable for wide-spread adoption among instructors. Buffardi & Edwards (2014) designed a system to encourage students' adherence to Test-Driven Development, and found that students more often start testing *late* than early in development, and test *after* writing solutions rather than before.

Smith et al. (2012) described their experience in incorporating peer testing into an honors data structure course without significantly reducing its heavy programming load. Liu, Kuo & Chen (2010) reported their experience in teaching metamorphic testing to various groups of students. Mao (2008) proposed a new teaching method, namely a Question-Driven Teaching Method (QDTM), to enrich the education for professional testers. Garousi (2010) designed a lab courseware for a testing course and reported the experiences gained from using the courseware. Software testing is a subject that can be difficult to teach, perhaps because it relies heavily on experiential learning (Smith et al., 2012). Therefore, instructors are continuously employing different methods and strategies in order to provide an effective delivery of the course.

2.2 Usage and benefits of different testing techniques

Harrison (2010) demonstrated that different techniques are needed for developer and tester testing, and stated that the double-headed testing project is a successful way for students to learn how to apply testing techniques used by both developers and testers. Yu et al. (2004) introduced a classification-tree method, or CTM, - a black-box testing method – in an undergraduate course. They concluded that initially white-box testing methods were more popular among students, who were later convinced of the benefits of CTM after they learned and used it. Software testers generally employ functional or specification-based testing. Black-box testing is more common than white-box testing in software industry (Chan et al., 2005; Garousi and Zhi, 2013; Ng et al., 2004). Chan et al. (2005) conducted a survey among software development organizations in the Asia-Pacific region, and noticed that a majority (91.2%) use functional or specification-based testing as opposed to structural testing (35.3%). Chen et al. (2014) studied the effectiveness, complementarity and impact of test case aspects of random and functional testing strategies, and observed that random testing is more effective in the early stage of testing on small applications, while functional testing is more scalable on large ones. Garousi and Zhi (2013) conducted a survey of software testing practices in Canada and found that manual testing is more dominant compared to automated testing.

2.3 Software testing and programming background

Initially, it was commonly believed that testers should not be required to have programming skills unless there is a need for automating tests since test automation is inherently a programming activity (Hendrickson, 2010). Yet, nowadays, the industry has recognized the role of effective programming skills for a successful tester. 80 percent of the quality assurance/test jobs advertised indicated knowledge of a specific language to be either a must or an advantage (Hendrickson, 2010). The role of programming language skills in software testing has also been accepted in the education of software testing, which requires experience in programming and, perhaps, something introductory students are not ready for until they have mastered other basic skills (Edwards, 2004).

2.4 Computer Science courses and gender

There are still fundamental gender disparities in computer use and careers in computer-related domains (Fleischmann et al., 2016). Women are perceived to be less skilled with computers than men (Cooper, 2006; Smith et al., 2005). They were also reported to possess lower computer skills (Christoph et al., 2015; Dickhauser and Stiensmeier-Pelster, 2003; Hargittai and Shafer, 2006; Sieverding and Koch, 2009), as well as lower computer self-efficacy, while expressing more negative attitudes toward computer-related domains than men (Christoph et al., 2015; Durndell and Haag, 2002; Li and Kirkup, 2007; Sun, 2008; Tomte and Hatlevik, 2011). Additionally, males are generally more interested and self-confident in the technical field of programming (Ertl and Helling, 2011).

Factors such as self-efficacy, self confidence, and pre-university experience influence success in computing and programming. Werner, Hanks, and McDowell (2004) found, through a survey of over 400,000 students entering freshman across the US, that the gender gap in computer use is almost non-existent but there is a very big confidence gender gap in computer skills. Female students are found to have lower levels of computer self-efficacy compared to males (Papastergiou, 2008). Guzman and Stanton (2009) supported that males reported higher self-efficacy with respect to the demands of IT occupation than females from their study of participants enrolled in undergraduate programs related to IT at two North East US institutions of higher education. It has been further argued that self-efficacy beliefs about computing may affect the degree of success, with which students accomplish computing tasks (Galpin et al., 2003). Several studies (Alvarado, Lee, & Gillespie, 2014; Askar and Davenport, 2009; Carter & Jenkins, 1999) concluded that female students are much less confident in their programming abilities than male students. In addition, Murphy et al. (2006) found that female students have less pre-college programming experience than their male peers. Students who had previous programming experiences tended to perform better than other students (Byrne and Lyons, 2001).

Sabitzer and Pasterk (2014) reported significant differences between male and female learning outcomes in an introductory programming course. Rubio et al. (2015) found some differences in learning between males and females in the traditional programming course although these differences were not statistically significant. On the other hand, Grant (2003) revealed that females outperformed males in a study of two programming courses, one procedural/structured and the other object-oriented.

Although female students are found to have lower self-efficacy beliefs and prior experience in computing abilities and programming, the effect of these factors on their success and performance in programming courses is still inconclusive. Therefore, it would be interesting to explore the effect of gender on success and performance in software testing.

3. RESEARCH METHODOLOGY

3.1 Course Details

Since the Systems Software Validation and Testing – called in the sequel 'Software testing' for short - is a relatively new course and not deeply-rooted in university curricula, it is of current concern to establish effective tools of knowledge control aimed to accomplish the education process. The course is included in the second year of undergraduate curriculum for the Bachelor of Engineering (Software Engineering). It is taught in a two-hour theory lecture and two-hour laboratory session every week together with exercises and assignments given at regular intervals.

3.2 Participants

Software Testing is a second-year must course within the curriculum of the Software Engineering Department. Consequently, the population under scrutiny comprises all students of the Department of Software Engineering at Atilim University, Ankara, Turkey. A sampling frame for the population is provided by the list of students registered for the course through the on-line information system of the University. Since 2010, when the program was first launched, on average, about 40 students register for this course each year. This study is based on the data collected over three years (between 2012 and 2014) from the participants.

Prior to this, students have studied such relevant disciplines as Introduction to Software Engineering, Software Requirements Engineering, and Human Computer Interaction. As for computer programming, this is not currently pre-requisite for Software Testing. Although C and C++ courses are offered in advance, the students who failed there can also enroll in Software Testing. Since C programming is a first-year course, there is a strong possibility that most of the students registered in Software Testing course have taken it before; whereas C++ is a second-year course. Additionally, there is a pre-requisite chain between C++ and C programming courses. It should be emphasized that, within the Software Testing domain, there are some aspects requiring programming skills. Therefore, it is crucial to indicate the likely effect of students' degree of programming knowledge on their achievements in Software Testing.

Since the language of education at Atilim University is English, the student body include not only native Turkish students, but also those from different countries, mostly developing countries, the former Soviet Union, various regions of Africa and the Middle East. The ages of the second- and third-year students typically range from 20 to 25 years. The majority of students in the Software Engineering Department are male: at the time of this research, female students constituted about 30% of the course participants. Nevertheless, the dynamics of the situation demonstrates that the percentage of female students has been increasing slowly but steadily. In this study, the data were collected for 35 female and 83 male students. The gender issue is included in this investigation because, as recent research (Fleischmann et al., 2016; Charles and Bradley, 2006) reveals, male employees outnumber females in the IT domain in general. Partially, this may be explained by the fact that, as a stereotype, computer-related occupations are mostly viewed as masculine careers (Cohoon and Aspray, 2006; Wajcman, 2000). The gender imbalance in IT enrolment and employment confirms that female students may be depriving themselves of potentially rewarding careers at an early age (Lang, 2012). In addition, female students in the software development area often tend to underestimate themselves, decreasing, in this way, both their motivation and chances for a good job. Hence, it is both necessary and overdue to observe the real situation concerning the comparative performances of both genders.

3.3 Procedure

Mainly, two approaches are being taught to students: structural and functional testing. In addition, automated testing as the next step of testing, is also taught. More specifically, different techniques of structural and functional testing are proposed one by one during two-hour theory lectures almost every week throughout the semester. Following these theory lectures, the related testing techniques are reinforced with the help of various exercises by favor of the two-hour laboratory sessions. The laboratory sessions last for 100 minutes. This length requires adjusting the difficulty level of the lab exercises in such a way that they can be solved by the students entirely within one session. For this reason, small program codes or functions on common topics that are easy to understand are preferred to encourage students to apply the testing technique effectively.

In general, the exercise questions that are offered to students in the laboratory have a small paragraph that specifies the goal of the program along with its code. In the questions, students are expected to apply the testing technique they have learned that week in the theory lecture. After applying the technique of that week, they find the test cases. To run these cases, CuTest Software is used as an open source testing tool for C codes. To put in another way, by running the test cases, students are in essence checking whether the expected output values they anticipate from the specification are the same as the actual output values detected once running the software for a specific set of input values. An example question of a regular lab is provided in the Appendix.

Regarding the evaluation of students' performances in this course, it is done in two ways: a traditional written exam (midterm and final) as well as a lab. exam (midterm and final). Students are not allowed to use computers in traditional written exams therefore focusing more on theoretical issues whereas computer usage is integral for lab. exams consequently covering both theoretical and the practical aspects of the course. Furthermore, in the weekly lab sessions, students apply the theory directly for practical problems. That is, given a program code, they use prescribed techniques to extract test cases. The responses are collected and graded for these weekly lab sessions. Besides, after covering all the structural testing techniques, students appeared in a lab exam that covered questions from each structural testing technique such as Basis path testing, data flow testing. This lab exam takes place in the mid of semester so it can be considered as lab midterm where all the topics related to the structural testing are covered. Similarly, after covering all the topics of functional testing a lab exam containing questions from

each functional testing techniques is conducted. In both lab exams, similar to each weekly lab exercises students are expected to use the testing tool to run their test cases that they develop by using the techniques. Here, the evaluation is based on deducing the correct test cases using a particular testing technique. Moreover, some exercises are given to check how good the students are using the testing tool and writing the test cases using test script. Students are free to apply any testing technique to deduce the test cases. The emphasis during evaluation is based on correctly writing test script for test cases. The course grading policy breakdown is: 70% for traditional written exams and 30% for lab sessions. In the lab, evaluation of students' performances on both weekly labs and lab exams constitutes this 30 %.
The students' programming background is determined by using their score in the programming course taken in the previous year/semester. If a student took both programming courses or a particular programming course more than once, then the score is calculated by taking the average of all scores.

3.4 Research Questions

This work was focused on providing answers to the research questions (**RQ**) formulated below.

RQ1. Is there a significant difference between the students' performances under different testing techniques?
RQ2. Does there exist a noticeable relationship between the tests results under different testing techniques?
RQ3. What is the importance of the programming background when applying different testing techniques?
RQ4. What is the influence of the gender factor on success in software testing assessments?
RQ5. How do various teaching strategies over the years affect the exam results in the software testing?

3.5 Data Analysis

In this part of the paper, the methods of statistical data analysis are employed (see, for example Siegel and Morgan, 1996). On the whole, the research methodology is based on a statistical approach and mostly hypotheses testing. Whenever the Z-test was applied, a pre-requisite examination of the data is also carried out. For example, tests for normality were performed using statistical software. The findings pertinent to the research questions 1-5 are supplied below along with the necessary comments.

RQ1. As reported by various studies from the industry, people generally prefer functional testing over structural testing. As such, the authors aim to investigate whether there is a relationship between performances in terms of structural, functional and automated testing.
To compare the students' performances in different testing techniques, the sign test with a significance level $\alpha=0.05$ is applied. Since three students were formally registered but, in fact, did not attend neither the classes nor the evaluation tests, they were excluded in the analysis. In total, the results for 115 individuals participating in all of the tests have been examined, and the following conclusions have been reached:
Out of 115 students, 65 were more successful in functional testing, while 47 in structural testing. The obtained z-value is -1.606, showing that there is no significant difference in the students' performance using the structural and functional testing techniques. The situation, though, becomes different for automated testing. Primarily, the grades on this test are significantly better

than those for both functional and structural testing techniques. Namely, in the juxtaposing of the results for structural and automated testing, 80 students demonstrated better performances in automated testing, while only 30 were better in structural testing. With the observed value z=-4.215, this shows convincingly that, in general, students are more successful in automated testing. Likewise, 73 out of 115 students demonstrated better performance in the automated test, while only 37 were so in functional testing. This yields an observed value of z=-3.337, implying that, on the whole, students perform better in automated tests.

RQ2. In this part of the study, the possible correlations between students' achievements in different testing techniques are analyzed. This is accomplished by finding the Spearman rank-correlation coefficient and by testing its significance. Generally speaking, such significance indicates that there exists a monotonicity relationship between the variables - either positive or negative. All tests have been performed at the level of significance α = 0.05. Table 1 demonstrates the values of Spearman's coefficient along with the corresponding P-values. Variables X and Y are the students' grades in the indicated testing techniques. As a result, it can be observed that a positive relationship exists between all testing techniques as shown in Table 1.

Table 1: Spearman's correlation coefficients and the corresponding P-values

i	X - Y	ρ_i	P – values p_i
1	structural testing – functional testing	0.290	0.0018
2	structural testing - automated test	0.348	0.0002
3	functional testing - automated test	0.208	0.0250

RQ3. As mentioned in the introduction, testing is an activity which requires a wealth of other skills, such as critical thinking and analytical and investigative skills. Since some testing techniques require abilities similar to those for computer programming, it appears necessary to find out whether there is an association between programming experience and different testing techniques. Table 2 represents the results pertinent to this matter. It can be observed that there exists a positive relationship between all of the variables, except programming background and functional testing techniques. As a result, the following conclusions can be reached: the stronger the student's programming background, the more successful he/she is in applying structural testing techniques and also in automated testing. Nevertheless, there is no evidence implying that computer programming skills are beneficial for learning functional testing techniques.

Table 2: Spearman's correlation coefficients and the corresponding P-values

i	X - Y	ρ_i	P – values p_i
4	Computer programming – structural testing	0.400	0.0000
5	Computer programming – functional testing	0.132	0.1649
6	Computer programming – automated test	0.299	0.0013

RQ4. Nowadays, ample research is taking place on gender-related issues within different disciplines. According to the literature, even though it is possible to find contradicting results, there is a common view that in programming courses males are more successful than females are. To put in another way, males are more inclined towards programming-type courses than females.

This study aims to investigate if there is any correlation between students' performance in software testing and the gender factor. Table 3 is a summary of the collected relevant data.

Table 3: Students' performance with respect to the gender factor
$\bar{x}$ = sample mean, s = sample standard deviation, n = sample size

i	Testing technique	Gender	$\bar{x}_i$	s_i	n_i
1	structural testing	female	47.17	19.80	35
2	structural testing	male	53.03	19.09	83
3	functional testing	female	55.99	15.33	35
4	functional testing	male	53.58	17.20	83
5	Automated test	female	59.53	21.54	35
6	Automated test	male	64.48	20.66	78

The following procedure has been applied: test the null hypothesis $H_0 : \mu_{female} = \mu_{male}$ against the alternative $H_1 : \mu_{female} \neq \mu_{male}$. As before, the level of significance is α=0.05. The following observed values of z–statistic are calculated 1.48, -0.75, and 1.14 for the structural, functional, and automated tests, respectively. This shows conclusively that in the framework of this research, there is no significant difference in the performance of male and female students in any type of testing.

RQ5. In order to determine the most effective way of training, the teaching strategies pertaining to different techniques during the three years subject to this study were purposely modified after 2012. While in 2012 only the hardcopy of the code had been distributed for developing test cases using structural testing techniques, both the hardcopy and the softcopy - that is, the facility to execute the buggy code - were provided during 2013 and 2014. To create test cases using functional testing, only the specifications to this end were provided in 2012; whereas in 2013 and 2014 a soft copy of the code was also provided in addition to those specifications. Furthermore, in 2013 and 2014 the students were introduced to automated testing simultaneously while teaching structural testing at the beginning of the semester. Meanwhile, in 2012, students started applying automated testing half-way through the semester, after they became familiar with creating test cases using various testing techniques. Therefore, it is important to investigate whether this change in the teaching strategy has had any affect on the students' performances in testing.
To answer these questions, the data were collected for 3 groups of software engineering students comprising 30, 46, and 42 individuals taking the course in the years 2012, 2013, and 2014, respectively. As stated before, since three students were formally registered but, in practice did not attend neither the classes nor the evaluation tests, they were not included in the analysis. The topics included in the tests were similar, while the modifications mentioned earlier had been incorporated into some applications of testing techniques. The collected data are displayed in Table 4.

Table 4: Students performance results (scores out of 100)
$\bar{x}$ = sample mean, s= sample standard deviation, n = sample size

i	year	testing technique	$\bar{x}_i$	s_i	n_i

1	2012	structural testing	62.12	18.55	30
2	2012	functional testing	50.65	15.42	30
3	2012	automated testing	56.95	19.70	30
4	2013	structural testing	50.11	18.46	46
5	2013	functional testing	55.44	14.47	46
6	2013	automated testing	65.70	22.74	43
7	2014	structural testing	44.81	18.10	42
8	2014	functional testing	55.69	19.52	42
9	2014	automated testing	65.50	17.02	42

Statistical tests were performed to determine whether there are significant differences among the students' performances in the three groups. All of the tests were conducted at the significance level $\alpha=0.05$, revealing the following conclusions concerning the population means: $\mu_1 > \mu_4$ and $\mu_1 > \mu_7$, while $\mu_4 = \mu_7$. This shows that, after the modifications made in the test assignments, the structural testing technique appears to be more difficult for students.

As for the functional testing techniques, it is derived that $\mu_2 = \mu_5 = \mu_8$, demonstrating that, on the whole, the students' performances in this test has not changed. Finally, after the first year, the afore-mentioned changes were made in the application of the automated testing. The statistical test shows that those changes were beneficial for the students as $\mu_3 < \mu_6 = \mu_9$.

4. RESULTS AND DISCUSSION

The results indicate that students outperformed in automated testing compared to structural and functional testing techniques. This might be because of the affinity with tools; students find automated testing easier, more interesting, and worthwhile compared to manual testing. Another reason might be that automated testing is the post-activity of other testing techniques where one simply uses the test cases derived previously using structural, functional, or other testing technique. They only need to be familiar with the testing tool and its environment in order to perform automated testing effectively. Moreover, the test functions available in the testing tools are generic in nature, and students merely need to utilize these functions according to the tested program. Once familiar with the automated testing environment, its script, how to write test cases, test suites etc., it does not require much extra effort to test different programs as students only need to know the test cases which have been already derived before using some testing techniques as well as the choice of appropriate test functions to test a given program.

It is also observed that there is a strong correlation in terms of success in automated, structural, and functional testing. It is common belief that testing activity is a compound activity, the aim is to test the working program by deriving test cases and, identifying related expected outputs, and consequently, to run test cases in order to reach expected outputs in the end. Even though each particular technique requires certain skills, the main premise behind these techniques is probing and investigating. Therefore, it is not surprising that if students are successful in one technique, they are successful in the others as well.

It is also established that, although there is an association between programming with structural and automated testing, success in programming and functional testing is not related. The relationship between programming with structural and automated testing is obvious considering both techniques are highly programming-oriented. Regarding the result related to functional testing, the authors started by suggesting that in functional testing, an ability to understand the logic is required and this logic also constitutes a base for programming. Therefore, one can expect a degree of correlation between functional testing and programming although not as strong as structural or automated testing, which are totally programming-oriented. After the analysis, it was concluded that there is no relation between the two - the reason for which can be that functional testing is not a code-oriented, but rather requirements' specification-based, technique. Therefore, it can be stated that knowledge of programming is not essential for success in functional testing.

It has been established through various studies that female students are less inclined to pursue computing and programming fields. There are a handful studies reporting female students' lack of success in computing and programming due to their low self-confidence and limited prior experience. The present study concludes that there is no difference in the performance of male and female students in testing. Murphy et al. (2006) established in their work about learning programming concepts that, while female subjects enter college having been introduced to and having mastered fewer concepts before college than males, they "catch up" with them in their introductory classes, reporting - at least retrospectively - having achieved similar mastery of concepts.

In the course of this study, changes in the teaching strategy affected the success of students negatively in structural testing in 2013 and 2014, where the testing tool was introduced along with structural testing at the beginning of the semester. One reason for this impact might be that students, being 2nd year undergraduates, could not handle two new concepts at the same time and merely focused on the one they found easier or interesting- in this case, automated testing - and it also reflected in their performance. The introduction of automated testing early in 2013 and 2014 resulted in a better performance by students as they naturally could spend more time and practice with the tool. At the same time, success in structural testing decreased in 2013 and 2014 compared to 2012, when only structural testing was introduced initially in the semester so that students could mainly focus on this topic alone.

Additionally, we thought that by providing the soft copy along with the hard copy while creating test cases in later years, we could improve the students' performance in structural testing. However, to our surprise, no improvement was seen when students were free to use the compiler (with soft copy) as it is indeed useful to catch the syntax errors, but one has to further use their analytic skills to find out about the semantic errors. A tester using structural testing techniques only needs to know in advance what a program does, i.e. its specifications, and must have access to the hard copy of the source code to create test cases using structural testing techniques, such as basis path testing, data flow testing, etc.

Changing the teaching strategy, i.e. providing the code with specifications, did not affect the success in functional testing. The reason behind this outcome can be that the availability of the code does not make any difference in test case generation, since in functional testing test case derivation is based on specifications alone.

5. PRACTICAL IMPLICATIONS

The present study has the following implications:

Students performed better in automated testing compared to manual testing, showing that the integration of this testing tool into the course makes it more appealing to the audience. It is commonly known that higher performances make students motivated. As expected, the performances of students in different testing techniques have a strong positive association.

Possibly, this is due to the fact that, though the considered testing techniques are based on specific skills, they all require similar abilities – i.e. probing and investigating – to detect errors.
Strong programming skills have a positive impact on the success in structural and automated testing, but not in functional testing. This implies that sufficient knowledge of programming is essential for this course and, as such, some pre-requisite programming courses should be offered in advance to ensure that the students' knowledge is up to the level to fulfill the requirements of this course.
Since there is no difference in performance in terms of gender, the authors believe that either female students already possess the same level of confidence and experience as their male counterparts at the beginning of the course; or if they lack self-confidence and/or prior experience, they eventually "catch up" with male students during the course, implying that no additional motivation is required for females.
It is beneficial to introduce only one concept at a time if this course is being taught in the initial years of the undergraduate program. The results of this study suggest that imparting students with the knowledge of both structural testing and automated testing lead to reduced performance in structural testing. Consequently, it would be better if automated testing were introduced a few weeks later than structural testing as this can provide the participants with the chance to solely focus on this topic before moving on to automated testing. This would have also provided them with enough time and practice on the latter. Another suggestion can be to encourage more practice to achieve better performance in structural testing as it is more detailed and requires additional efforts.
Structural testing - e.g. basis-path testing or data-flow testing - requires a hard copy and specifications of the corresponding program to create test cases; while only specifications are required to create test cases for functional testing. Providing additional material - in this case, a soft copy of the program - made no positive difference in performance within this study. It can be said that additional material does not help to achieve better testing as it seems that either the students did not use it, or they did not receive additional clues for detecting errors. This might have happened because either they were unaware of such additional clues they could derive from the material or they could not manage to grasp them because their attention was divided or lost due to the presence of multiple material.

6. CONCLUSION, LIMITATIONS AND FUTURE WORK

These findings are based on the analysis conducted using three years of data collected while teaching a course in software testing. In general, students who are successful in a particular testing technique are found to be strong in other testing techniques as well. They generally performed better in automated testing compared to manual testing. It can also be concluded that strong programming background can positively affect success in the Software Testing course. Additionally, there is no difference between the performance of male and female participants. Moreover, it is advisable to introduce teaching concepts one at a time because students find it difficult to grasp the ideas otherwise.
Obviously, there are some limitations to this study. For example, student's strength in programming is calculated using the score of C programming courses taken in previous year/semester. Such scores may not reflect their current level of programming knowledge. There is a possibility for one to strengthen programming abilities by receiving private tuition or undertaking project work. Also, it is equally likely that students' knowledge might have diminished due to a lack of practice since last taking programming course. Furthermore, attempt

was made to ensure that the exercises given for different testing techniques have similar difficulty level to guarantee that the difference in success between these testing techniques is due to the inherent complexity of the technique itself and not because of different exercises. Still, there is small probability that a certain degree of change in success may be due to the difference in the difficulty levels of the exercises. This can be handled by giving the same exercise to be solved using different testing techniques. Naturally, it might be difficult to find one exercise suitable for different testing techniques. Besides, students may lose interest and motivation if the same problem is repeated while teaching different testing techniques.
As such, it is obviously premature to consider the present results as final since there is a lack of similar type of studies, with which the authors can compare their results. More work needs to be done in different settings to draw sound conclusions in this respect. This study can be extended using data from other institutions for confirmation. In 2015, this course was moved to the third year in the curriculum so that most of the students can gain experience not only in C programming, but also in object-oriented programming languages like C++ in advance. Additionally, they can brush up their skills in industry during the summer practice at the end of the second year. In the future, other research can be conducted to see how this change affects success in testing courses. Due to this change, another testing tool, JUnit, has been introduced in the course on top of the CuTest priorly in use. In the past, second-year students taking this course mostly had knowledge of structured programming, i.e. C, and, therefore, only CuTest was used in this course. Third-year students also have knowledge of object-oriented programming and can test their C++ programs with the help of JUnit. A comparison can be conducted as to the ease of use between CuTest and JUnit because the latter is considered a better tool due to more features and better interface.

Acknowledgement
The authors would like to express their sincere gratitude to the anonymous referees and the review manager for their constructive remarks, which helped to improve the presentation of this work.

References

1. Alvarado, C., Lee, C. B., and Gillespie, G., 2014. New CS1 pedagogies and curriculum, the same success factors? In Proceedings of the 45th ACM technical symposium on computer science education (pp. 379-384) New York, NY, USA: ACM.
2. Askar, P., and Davenport, D., 2009. An investigation of factors related to self-efficacy for Java programming among engineering students. The Turkish Online Journal of Educational Technology, 8(1), 26–32.
3. Barbosa, E.F., Silva, M.A.G., Corte, C.K.D, and Maldonado, J.C., 2008. Integrated teaching of programming foundations and software testing. In 38th Frontiers in Education Conference, Saratoga Springs, NY, Oct. 2008. 6p.
4. Buffardi, K. and Edwards, S.H., 2014. A formative study of influences on student testing behaviors. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 597-602.
5. Byrne, P., and Lyons, G. 2001. The effect of student attributes on success in programming. In Proceedings of the 6th annual conference on Innovation and technology in computer science education (ITiCSE '01). ACM, New York, NY, USA, 49-52.
6. Carter, J., and Jenkins, T., 1999. Gender and programming: what's going on? SIGCSE Bulletin, 31(3), 1e4. http://dx.doi.org/10.1145/384267.305824.

7. Charles, M., and Bradley, K., 2006. A matter of degrees: female underrepresentation in computer science programs cross-nationally. in Aspray, J.M.C.a.W. (Ed.), Women and Information Technology, MIT Press, Cambridge, MA, pp. 183-203.
8. Christoph, G., Goldhammer, F., Zylka, J., and Hartig, J., 2015. Adolescents' computer performance: the role of self-concept and motivational aspects. Computers & Education, 81, 1-12.
9. Cohoon, J.M., and Aspray, W., 2006. A critical review of the research on women's participation in postsecondary computing education. in Aspray, W. and McGrath Cohoon, J. (Eds), Women and Information Technology: Research on Underrepresentation, The MIT Press, Cambridge, pp. 137-80.
10. Chan, F.T., Tse, T.H., Tang, W.H., Chen, T.Y., 2005. Software Testing Education and Training in Hong Kong. In Proceedings of the Fifth International Conference on Quality Software (QSIC '05). IEEE Computer Society, Washington, DC, USA, 313-316.
11. Chen, Z., Memon, A., and Luo, B., 2014. Combining Research and Education of Software Testing: A Preliminary Study. In Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). ACM, New York, NY, USA, 1179-1180.
12. Cooper, J., 2006. The digital divide: the special case of gender. Journal of Computer Assisted Learning, 22 (5), 320-334.
13. Cowling, T. 2012. Stages in teaching software testing. In Proceedings of the 34th International Conference on Software Engineering (ICSE '12). IEEE Press, Piscataway, NJ, USA, 1185-1194.
14. Dickhauser, O., and Stiensmeier-Pelster, J., 2003. Gender differences in the choice of computer courses: applying an expectancy-value model. Social Psychology of Education, 6 (3), 173-189.
15. Durndell, A., and Haag, Z., 2002. Computer self efficacy, computer anxiety, attitudes towards the Internet and reported experience with the Internet, by gender, in an East European sample. Computers in Human Behavior, 18 (5), 521-535.
16. Edwards, S.H., 2004. Using software testing to move students from trial-and-error to reflection-in-action. In Proceedings of the 35th SIGCSE technical symposium on Computer science education (SIGCSE '04). ACM, New York, NY, USA, 26-30.
17. Elbaum, S., Person, S., Dokulil, J., Jorde, M., 2007. 'Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable,' 29th International Conference on Software Engineering (ICSE'07), IEEE Computer Society, 2007, pp. 688-697.
18. Ertl, B., and Helling, K., 2011. Promoting Gender Equality in Digital Literacy. Journal of Educational Computing Research. 45 (4), pp. 477-503.
19. Fleischmann, A., Sieverding, M., Hespenheide, U., Weiß, M., and Koch. S.C., 2016. See feminine - Think incompetent? The effects of a feminine outfit on the evaluation of women's computer competence. *Comput. Educ.* 95, C (April 2016), 63-74.
20. Galpin, V., Sanders, I., Turner, H., Venter, B., 2003. Computer self-efficacy, gender, and educational background in South Africa, Technology and Society Magazine, IEEE , vol.22, no.3, pp.43,48, Fall 2003.
21. Garousi, V. and Zhi, J., 2013. A survey of software testing practices in Canada. J. Syst. Softw. 86, 5 (May 2013), 1354-1376.
22. Garousi, V., 2010. 'An Open Modern Software Testing Laboratory Courseware - An Experience Report', In Proceedings of the 2010 23rd IEEE Conference on Software Engineering Education and Training (CSEET '10). IEEE Computer Society, Washington, DC, USA, 177-184.

23. Grant, N.S., 2003. A study on critical thinking, cognitive learning style, and gender in various information science programming classes. In *Proceedings of the 4th conference on Information technology curriculum* (CITC4 '03). ACM, New York, NY, USA, 96-99.
24. Guzman, I.R., and Stanton, J.M., 2009. IT occupational culture: the cultural fit and commitment of new information technologists. Information Technology & People, Vol. 22 Iss 2 pp. 157 – 187.
25. Hargittai, E., and Shafer, S., 2006. Differences in actual and perceived online skills: the role of gender. Social Science Quarterly, 87 (2), 432-448.
26. Harrison, N.B., 2010. Teaching software testing from two viewpoints. J. Comput. Small Coll. 26, 2 (December 2010), 55-62.
27. Hendrickson, E., 2010. Do testers have to write code? Available at testobsessed.com/2010/10/testers-code/
28. Lang, C., 2012. Sequential attrition of secondary school student interest in IT courses and careers. Information Technology & People, Vol. 25 Iss 3 pp. 281 – 299
29. Li, N., and Kirkup, G., 2007. Gender and cultural differences in Internet use: a study of China and the UK. Computers & Education, 48 (2), 301-317.
30. Liu, H., Kuo, F.C. and Chen, T.Y., 2010. Teaching an End-User Testing Methodology. In Proceedings of the 2010 23rd IEEE Conference on Software Engineering Education and Training (CSEET '10). IEEE Computer Society, Washington, DC, USA, 81-88.
31. Mao, C., 2008. ‘Towards a Question-Driven Teaching Method for Software Testing Course’, In Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 05 (CSSE '08), Vol. 5. IEEE Computer Society, 645-648.
32. Murphy, L., Richards, B., McCauley, R., Morrison, B.B., Westbrook, S., and Fossum, T., 2006. Women catch up: gender differences in learning programming concepts. SIGCSE Bull. 38, 1 (March 2006), 17-21
33. Ng, S. P., Murnane, T., Reed, K., Grant, D. and Chen, T.Y., 2004. A Preliminary Survey on Software Testing Practices in Australia. In Proceedings of the 2004 Australian Software Engineering Conference (ASWEC '04). IEEE Computer Society, Washington, DC, USA, 116-.
34. Papastergiou, M., 2008. Are Computer Science and Information Technology still masculine fields? High school students’ perceptions and career choices, Computers & Education, Volume 51, Issue 2, September 2008, Pages 594-608.
35. Rubio, M.A., Romero-Zaliz, R., Mañoso, C., and de Madrid. A.P., 2015. Closing the gender gap in an introductory programming course. *Comput. Educ.* 82, C (March 2015), 409-420.
36. Sabitzer, B., and Pasterk, S., 2014. Brain-based programming continued. In Frontiers in education conference, 2014 IEEE, 1495-1500.
37. Siegel, A.F., and Morgan, C.J., 1996. Statistics and Data Analysis: An Introduction, John Wiley & sons, Inc., 2nd edition, 1996.
38. Sieverding, M., and Koch, S.C., 2009. (Self-)Evaluation of computer competence: how gender matters. Computers & Education, 52 (3), 696-701.
39. Smith, J. L., Morgan, C. L., and White, P.H., 2005. Investigating a measure of computer technology domain identification: a tool for understanding gender differences and stereotypes. Educational and Psychological Measurement, 65 (2), 336-355.
40. Smith, J., Tessler, J., Kramer, E., and Lin, C., 2012. Using peer review to teach software testing. In Proceedings of the ninth annual international conference on International computing education research (ICER '12). ACM, New York, NY, USA, 93-98.

41. Sun, S., 2008. An examination of disposition, motivation, and involvement in the new technology context computers in human behavior. Computers in Human Behavior, 24 (6), 2723-2740.
42. Timoney, J., Brown, S., Deshi, Ye., 2008. ‘Experiences in Software Testing Education: Some Observations from an International Cooperation’, Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for, IEEE Computer Society, pp.2686-2691.
43. Tomte, C., and Hatlevik, O.E., 2011. Gender-differences in self-efficacy ICT related to various ICT-user profiles in Finland and Norway. How do self-efficacy, gender and ICT-user profiles relate to findings from PISA 2006. Computers & Education, 57 (1), 1416-1424.
44. Wajcman, J., 2000. Reflections on gender and technology studies: in what state is the art? Social Studies of Science, Vol. 30 No. 3, pp. 447-64.
45. Werner, L.L., Hanks, B., and McDowell, C., 2004. Pair-programming helps female computer science students. Journal on Educational Resources in Computing, 4(1). http://dx.doi.org/10.1145/1060071.1060075.
46. Yu, Y.T, Ng, S.P., Poon, P.L., Chen, T.Y., 2004. On the testing methods used by beginning software testers, Information and Software Technology, Volume 46, Issue 5, 15 April 2004, Pages 329-335