Confirmation Bias in Software Development and Testing: An Analysis of the Effects of Company Size, Experience and Reasoning Skills

Gul Calikli
University of Glasgow
37 PUBLICATIONS **355** CITATIONS
SEE PROFILE

Berna A. Uzundag
Kadir Has University
22 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Ayse Bener
Ryerson University
181 PUBLICATIONS **3,812** CITATIONS
SEE PROFILE

Confirmation Bias in Software Development and Testing: An Analysis of the Effects of Company Size, Experience and Reasoning Skills

Gul Calikli

Department of Computer Engineering, Software Research Laboratory, Bogazici University, Turkey
gul.calikli@boun.edu.tr

Berna Arslan

Department of Computer Engineering, Software Research Laboratory, Bogazici University, Turkey
berna.arslan@boun.edu.tr

Ayse Bener

Department of Computer Engineering, Software Research Laboratory, Bogazici University, Turkey
bener@boun.edu.tr

Keywords: POP-II.B. problem comprehension, testing; POP-V.A. confirmation bias, cognitive bias

Abstract

During all levels of software testing, the goal should be to fail the code to discover software defects and hence to increase software quality. However, software developers and testers are more likely to choose positive tests rather than negative ones. This is due to the phenomenon called confirmation bias which is defined as the tendency to verify one's own hypotheses rather than trying to refute them. In this work, we aimed at identifying the factors that may affect confirmation bias levels of software developers and testers. We have investigated the effects of company size, experience and reasoning skills on bias levels. We prepared pen-and-paper and interactive tests based on two tasks from cognitive psychology literature. During pen-and-paper test, subjects had to test given hypotheses, whereas interactive test required both hypotheses generation and testing. These tests were conducted on employees of one large scale telecommunications company, three small and medium scale software companies and graduate computer engineering students resulting in a total of eighty-eight subjects. Results showed regardless of experience and company size, abilities such as logical reasoning and strategic hypotheses testing are differentiating factors in low confirmation bias levels. Therefore, education and/or training programs that emphasize mathematical reasoning techniques are useful towards production of high quality software. Moreover, in order to investigate the relationship between code defect density and confirmation bias of software developers, we performed an analysis among developers who are involved with a software project in a large scale telecommunications company. We also analyzed the effect of confirmation bias during software testing phase. Our results showed that there is a direct correlation between confirmation bias and defect proneness of the code.

1. Introduction

Human aspects are one of the basic components of software development and testing. Cognitive biases belong to these aspects and they are defined as the deviation of the human mind from the laws of logic and accuracy (Stacy & MacMillan, 1993). The notion of cognitive biases was first introduced by Tversky and Kahneman (1971) and further elaborated to comprise various bias categories (Kahneman, Slovic, & Tversky, 1982**.** Availability, anchoring and representativeness are examples of cognitive biases.

As far as we know, Stacy and MacMillan (1993) are the two pioneers who recognized the possible effects of cognitive biases on software engineering. In this study, we have investigated the confirmation bias, which is defined as the tendency of people to seek for evidence that could verify their hypotheses rather than refuting them. The term confirmation bias was first used by P.C. Wason in his rule discovery experiment (Wason, 1960).

The importance of confirmation bias in software engineering comes from the fact that, most of the defects are overlooked unless the goal is to fail the code during all levels of software testing. In other words, high confirmation bias levels of software developers and testers lead to an increase in software defect density which in turn results in a decrease in software quality. Empirical evidence shows that testers are more likely to choose positive tests rather than the negative ones (Teasley, Leventhal, & Rohlman, 1993). However, during all levels of software testing the attempt should be to fail the code. This study is concentrated on factors affecting confirmation bias, since circumvention of the effects of confirmation bias requires that we know about these factors. In this study, we also perform a small scale empirical analysis about the effects of confirmation bias on software development and testing.

We propose a method based on Wason's work to quantify confirmation bias levels. Quantification of confirmation bias helps us to analyze the effect of confirmation bias on software defect density as well as analyzing potential factors that may affect confirmation bias. In the following section, we will give more detailed information about confirmation bias. Then we will explain our methodology in detail. Finally, results will be presented, discussed together with threats to validity and potential future directions will be given.

2. Confirmation Bias

The term confirmation bias was first used by Peter C. Wason in his rule discovery experiment, where the subject must try to refute his/her hypotheses to arrive at a correct solution (Wason, 1960). Wason also explained the results of his selection task experiment using facts based on confirmation bias (Wason, 1968). This section explains these two experiments.

2.1. Wason's Rule Discovery Task

In this experiment, Wason asked his subjects to discover a simple rule about triples of numbers. The experimental procedure can be explained as follows: Initially, subjects are given a record sheet on which the triple "2 4 6" is written. The subject is told that "2 4 6" conforms to a simple rule which is only known by the experimenter at the beginning of the experiment. In order to discover the rule, the experimenter asks the subject to write down triples together with the reasons of his/her choice on the record sheet. After each instance, the experimenter tells whether the instance conforms to the rule or not. The subject can announce the rule only when he/she is highly confident. If the subject cannot discover the rule, he/she can continue giving instances together with reasons for his/her choice. This procedure continues iteratively until either the subject discovers the rule or he/she wishes to give up. If the subject cannot discover the rule in 45 minutes, the experimenter aborts the test.

2.2. Wason's Selection Task

In Wason's original task, the subject is presented with four cards, which have a letter on one side and a number on the other. These cards are placed on a table showing D, K, 3 and 7 respectively. The subject is then given the hypothesis "If a *card has a D on one side, then it has a 3 on the other side*" and he/she is asked which card(s) he/she should turn over to test whether the given hypothesis is true or false regarding the four cards presented to him/her. The hypothesis can be considered as a "*If P, then Q*" sentence. The correct way to test this hypothesis would be to select the *P* and *not Q* cards, which corresponds to selecting *D* and *7* respectively.

3. Methodology

Our methodology aims to quantify confirmation bias levels in order to make empirical analyses to investigate the factors affecting confirmation bias.

3.1. Preparation of the Tests

Pen-and-Paper Test. We prepared the pen-and-paper test based on Wason's Selection Task (Wason, 1968). Our test consisted of two parts including twenty-two questions of three different types. There were eight abstract, six thematic questions in the first part, whereas the second part contained eight questions about software domain.

Abstract questions where subject is asked to check the validity of hypothesis of the form "*If P then Q"*, can be answered correctly by pure logical reasoning. However, a subject might select the cards *D* and *3*, when he/she is asked to check the validity of the hypothesis "*If a card has a D on one side, then it has a 3 on the other side.*" Regarding the four cards presented to him/her. Such behavior of the subject can be explained by one of the following reasons:

1. *Verifying:* Subject's aim might be to *verify* the given hypothesis.
2. *Matching:* Subject might select the cards just by *matching* the letter *D* and number *3*. In other words, any reasoning strategy is not employed by the subject. This phenomenon, which was first defined by Evans (1972), is called *matching bias.*

One of the abstract questions was the question proposed by Wason in his selection task. Table 1 shows all versions of the original question with their possible answers categorized as true/false antecedent and true/false consequent. In order to analyze response strategies of subjects to these questions, all three negated versions of this original question were included in the test. Three of the remaining abstract questions had slight variations from the original Wason's Selection Task question.

Rule	TA	FA	TC	FC
(If P, then Q) If there is a D on one side, then there is a 3 on the other side	D	K	3	7
(If P, then not Q) If there is a D on one side, then there is not a 3 on the other side	D	K	7	3
(If not P, then Q) If there is not a D on one side, then there is a 3 on the other side	K	D	3	7
(If not P, then not Q) If there is not a D on one side, then there is not a 3 on the other side	K	D	7	3

Table 1 – The four logical choices in Wason's selection task with negated components permuted, TA: True Antecedent, FA: False Antecedent, TC: True Consequent, FC: False Consequent

Each thematic question in the test represented a probable real-life situation, so that they could be answered correctly using the cues produced by memory. Finally, domain-specific questions required participants to analyze a software problem which is independent of programming tools and environment.

Interactive Test. This test was a replication of Wason's Rule Discovery Experiment. Hence, similar to the original experiment, the experimenter performs the test face-to-face with each subject one at a time. Prior to the test, the experimenter explains to each subject the experimental procedure and the subject is also told that there is no time constraint.

3.2. Evaluating Pen-and-Paper Test

Falsifier/Verifier/Matcher Classification. Given the conditional rule of the form *if P, then Q*, the subject who selects *P, Q* as the answer can either be a verifier or matcher. Similarly, the same answer for the rule *if P, then not-Q*, means that the subject can be a falsifier or a matcher. In order, to overcome this fuzziness, we employed the method of Reich and Ruth (1982), which is explained below as follows:

- choice of not-Q in the rule "If P, then Q" = falsifying
- choice of not-Q in the rule "If P, then not Q" = verifying
- choice of P in the rule "If not P, then Q" = matching
- choice of not-Q in the rule "If not-P, then Q" = falsifying
- choice of P in the rule "If not P, then not Q" = matching

- choice of not-Q in the rule "If not P, then not Q" = verifying

According to the responses given to the four types of an abstract question, the subject was categorized as a falsifier, verifier or a matcher. This categorization decision was given according to the majority of the response tendencies, i.e. if a subject usually responded in a falsifying strategy, then he/she was categorized as a falsifier. There was also the possibility that the subject could not be categorized. This categorization neglects a large proportion of data provided by the subjects. On the other hand, it gives a general view about the subjects' responses. For these reasons, we used this method and labelled subjects, whom we could not classify, as *None*.

Insights. As an additional measure of the test, we took insights of the subjects into account. Johnson-Laird and Wason (as cited in Evans, Newstead & Bryne, 1993) proposed that subjects were in one of three states of insight as follows:

No insight. Subjects attempted to verify and chose *p* alone or the combination of *p* and *q* in the original question.

Partial insight. Subjects attempted to both verify and falsify the rule and hence chose *p*, *q* and *not-q* cards in the original question.

Complete insight. Subjects only attempted to falsify the rule and chose *p* and *not-q* cards in the original question.

Since we had more than one question in the test, we devised a method to classify the subjects according to this concept. Subjects who answered all of the abstract questions correctly were classified to have *complete insight*. Those who chose true antecedent alone or the combination of true antecedent and true consequent in more than or equal to half of the abstract questions were classified as having *no insight*. Finally, subjects who chose the combination of true antecedent, true consequent and false consequent in more than or equal to the half of the questions were classified as having *partial insight*.

3.3 Evaluating Interactive Test

Eliminative/Enumerative Index. This index aims to give an idea about how a subject thinks while he/she is going forward in the interactive test. A subject may think more eliminative and test more hypotheses or may think more enumerative and test similar hypotheses with different instances. The calculation of this index takes into account the nature of the instances in relation to their reasons for choice. The index is calculated as a ratio of the number of subsequent instances incompatible with each reason of choice to the number of compatible instances, summed over all proposed reasons. It is desirable to have this index to be greater than 1. Wason indicates that when this value is greater than 1 (the higher the better), the less confirmation bias of the subject is.

Test Severity. Severe testing consists of testing observations that have a high probability of being true in focal hypothesis and a low probability under all possible hypotheses (Poletiek, 2001). The severity of a test can be thought of as the power to eliminate alternative hypotheses. Poletiek's rule discovery experiment (2001) presented the subjects with the triple "2 7 6" and asked them to discover the rule that this triple conformed to. Test severity was calculated for each subject as follows:

$$S(x,H,b) = \frac{P(x \mid H,b)}{P(x \mid b)} \qquad (1)$$

This formula was defined by Popper (Poletiek, 2001), and *x* represents the severity of a test, *H* represents the hypothesis and *b* stands for the background knowledge. The severity of a test *x* is interpreted as the supporting evidence of the theory *H* given the background knowledge *b*. A test is more severe when the chance of the supporting observation occurring under the assumption of the hypothesis *H* exceeds the chance of its occurring without the assumption of the *H* (i.e. with the assumption of the background knowledge *b* only). The higher this ratio is (exceeds 1), the higher the severity of the test is.

To carry out a similar calculation, we have defined a set of possible alternative hypotheses of the interactive test as shown in Table 2. Instances given by the subjects are categorized as positive or negative tests according to their compliance with the experimenter's rule. Then, a positive test is more severe if it *excludes* more alternative hypotheses shown in Table 2 and a negative test is more severe if it *includes* more alternative hypotheses. For each instance given by the subject, a severity score was calculated. Finally, a mean severity score for each subject was calculated over all of his/her instances. This mean score could be between 0 and 27, since there were twenty-seven alternatives defined.

Poletiek (2001) discussed that maximizing the severity of tests may be a successful strategy if the subject has enough confidence in his/her hypothesis. Starting with a severe test may not result in more knowledge, so that it may be a better strategy to start with a mild test, increasing its severity when there is more evidence and then decreasing its severity again when one becomes quite sure about the validity of the hypothesis.

1	Integers ascending with increments of 2	15	Sum of the first and second integer is the third integer
2	Integers ascending with increments of k, where k = 1,2, ...	16	The triples of the form (2n 4n 6n), where n = 1,2,3, …
3	Three integers in ascending order such that the average of the first and third integer is the second integer.	17	The triples of the form (n 2n 3n), where n = 1,2,3, …
4	The average of the first and third integer is the second integer	18	Second integer is greater than the first one
5	Even integers ascending with increments of 2	19	Third integer is greater than the first integer
6	Integers ascending with increments of m = 2k, where k = 1,2,3, …	20	Difference between the third and the first integer is even
7	Integers ascending or descending with increments of m = 2k, where k = 1,2,3, …	21	Greatest common divisor (GCD) of the integers is 2
8	Even integers in ascending order	22	Ascending integers such that each integer is 1 less than a prime number
9	Positive even integers in ascending order	23	Any three rational numbers
10	Three even integers in any order	24	Positive real numbers in increasing order
11	Three integers in any order, none of them are identical	25	Positive integers in increasing order
12	Three integers in any order, two or three of them are identical	26	Three integers whose sum is even
13	Three integers in ascending order such that difference between third and first number is even	27	Three even integers greater than zero
14	Integers ascending or descending with increments of k, where k = 1, 2, 3, …		

Table 2 – The Set of Plausible Alternative Hypotheses

4. Experiment

4.1. Participants

Participants were employees of four companies and graduate computer engineering students. One of the companies was a large scale telecommunications company C_1 (11 females, 23 males, mean age = 29.06 years). There were three other small and medium scale software companies C_2 (6 males, mean age = 24 years), C_3 (1 female, 7 males, mean age = 29.63 years), and C_4 (1 female, 11 males, mean age = 27.17 years). In addition, twenty-eight graduate computer engineering students participated in the study (7 females, 21 males, mean age = 27.96 years).

Most of our subjects had a bachelor degree in computer science, mathematics or other engineering fields.

4.2. Procedure

All participants from companies were tested in their work environment. Students were tested at the university. First, it was explained that the tests do not aim at measuring IQ or similar capabilities. It has been told that the goal of the tests was identifying how people think. Subjects were asked to fill in the form about their personal information. Information about age, gender, B.S. field and university, M.S./M.A. field and university (if they exist), software development experience (in years), software testing experience (in years) was taken from each participant.

In each company, pen-and-paper test was applied to the employees as a group after the explanation of the test. Later, subjects participated one by one in the interactive test. Durations of both tests were recorded.

5. Results

In the following subsections, results of the analyses of the effects of company size, experience and reasoning skills are presented.

5.1. Effects of Company Size, Experience and Reasoning Skills on Confirmation Bias

The Analysis of Company Size. In this part, results of the large scale telecommunication company are compared with the results of three small/medium scale software companies in terms of both test performances.

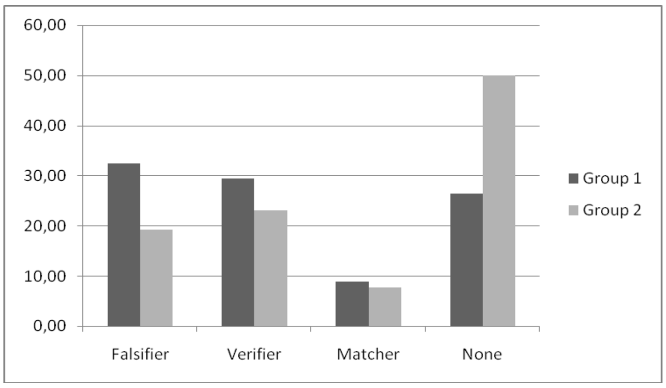

Figure 1 – The Distribution of Falsifiers, Verifiers and Matchers within Group 1(large scale company) and Group 2(small and medium scale companies)

In Figure 1, the percentages of falsifiers, verifiers and matchers for both groups can be seen. The large scale company, which is denoted as Group 1 in Figure 1, has a higher percentage of falsifiers, verifiers and matchers. Since this distribution alone is not very explanatory due to the high percentage of subjects not categorized, we examined another aspect, namely ‘insights’. The distribution of insights is shown in the Figure 2. According to Figure 2, the large company had a higher percentage of subjects with complete insight, slightly smaller percentage of subjects with partial insight and smaller percentage of subjects with no insight. But this difference is not significant statistically. Also, no significant difference was observed in the scores of the abstract questions indicating that no difference in logical reasoning skills was observed among the two groups.

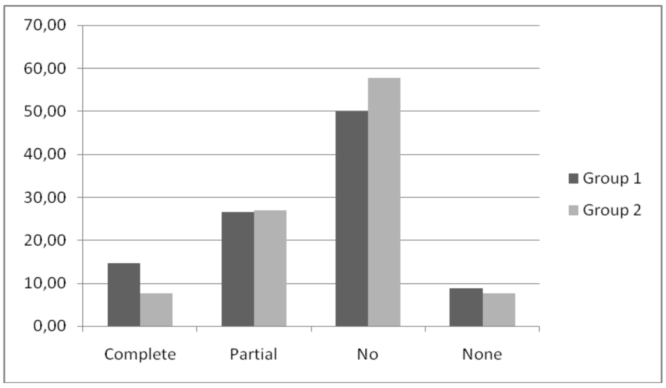

Figure 2 – The Distribution of Subjects according to their Insights within Group 1(large scale company) and Group 2(small and medium scale companies)

However, analyzing the interactive tests yielded differences between two groups. After excluding outliers from both groups, Mann-Whitney U-test was used to compare the ranks for eliminative/enumerative index values. The results of the test were significant, $z = -2.76$, $p < .01$. Group 1 (n=34) had an average rank of 35.47, while Group 2 (n=26) had an average rank of 23.06. This indicated that employees of the large scale company performed better in terms of elimination of their hypotheses.

Another Mann-Whitney U-test was used to compare the ranks for test severity values among Group 1 and Group 2. The results of the test were significant, $z = -2.26$, $p < .05$. Group 1 had an average rank of 34.96, while Group 2 had an average rank of 24.67. This indicated that, Group 1 had higher test severity values indicating a strategy that employed high severe testing. But, as mentioned before, it is a successful strategy to start with a mild test, continue with a more severe test and then end with a mild test again. In order to compare these severity strategies among two groups, we have made use of Vincent curves (as cited in Hilgard, 1938). These curves can be used to visualize the change in test severity of a group of subjects until the discovery of the correct rule. We have used the original method proposed as follows:

A number *N* was taken to be the bin size and total number of instances given by each subject was divided into fractions according to this number. Within each fraction, the average of the test severities of instances in that fraction was calculated for each subject. Then, all severity values in each fraction were averaged to give the mean severity value of all instances given by subjects in that fraction. As N, we have used the smallest number of instances given within the group of subjects. For instance, the division of 13 instances into N=4 fractions would be 4, 3, 3, 3. Additional instances that were left over from the division were added to the beginning according to the original procedure. Then, in the first fraction the average severity value of the four instances that fell in that fraction was calculated. This procedure was repeated for all subjects until mean severity values for N bins were obtained. A Vincent curve depicts these N data points, and it can be used to interpret the severity strategy employed by the subject.

Figure 3 shows the Vincent curve for the test severities of the three groups of subjects. The bin size was taken to be three, equal to the minimum number of instances given by a subject.

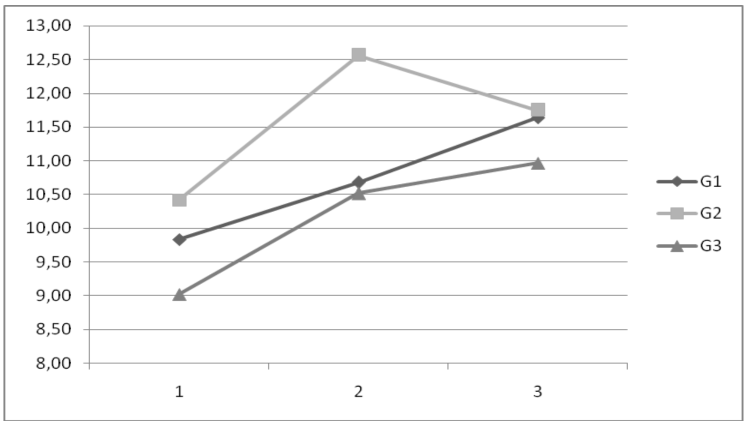

Figure 3 – Vincent Curve showing Test Severity Strategy of three groups, G1 being the group of small and medium scale companies, G2 being the large scale company and G3 being the group of graduate students

Examining the curve for the large scale company, denoted as G2 in the figure, shows that testing started with a mild test and became more severe and then ended in a milder test. This strategy was mentioned to be a successful strategy for hypothesis testing. The curve denoted with G1 in the figure shows that the overall strategy of the small and medium scale companies was to begin with a mild test and gradually enhance the severity. By comparing these two curves, we can say that the strategy of the large scale company shows a more sensible hypothesis testing strategy.

However, Poletiek (2001) also mentioned that selecting severe rather than weak tests does not necessarily reflect a motivation to falsify. In fact, the purpose of such a strategy might be to give a strong proof for one’s theory. To investigate that, we have analyzed the percentages of subjects in both groups, who have showed strong confirmation tendencies by repeating or reformulating their reasons for choice or their rules and who have immediately announced new rules without giving an instance, i.e. without testing their hypotheses. We have determined the percentages of these subjects within all groups as shown in Table 3. A subject who may have repeated a reason may also have made an immediate rule announcement, so that it is not the case that a subject can only be found in one cell of Table 3.

In order to compare these statistics between groups, we have merged the columns of Table 3 and compared the number of subjects who behaved in a confirmative way between groups, i.e. subjects who engaged in at least one activity defined in the columns. We have found no significant difference between the groups of companies according to company size, ($\chi^2(1) = .09$, $p > .7$). Hence, we can say that large scale company engaged in a better hypothesis testing strategy by looking at eliminative/enumerative index, test severity values and Vincent curves.

The Analysis of Experience. We have investigated whether there was a significant correlation between software testing experience and test severity. All subjects who had software testing experience were included in the test (n=27), and no significant result was obtained. Also, no significant correlation was found between software testing experience and eliminative/enumerative index.

A similar analysis was conducted for software development experience. All subjects who had software development experience were included in the test (n=50). No significant correlation was found between software development experience and eliminative/enumerative index as well as test severity.

Hence, results showed no effect of software testing or development experience on the ability of hypothesis testing.

Another analysis was conducted to compare test results among subjects taken from graduate students and employees of all companies, where the subject selection criterion was having the combination of software development and testing experience greater than the average number of years of experience found among graduate students. The average experience value was taken from the students’ group to make sure that there will be enough subjects from this group in the analysis.

Mann-Whitney U-test was used to compare the ranks for test severity, eliminative/enumerative index and test durations among employees (n=43) and students (n=13). The only significant result of the tests showed that interactive test duration of the student group was lower, z = -1.39, p < .05. First group had an average rank of 30.15, while second group had an average rank of 23.04. Hence, we concluded that the group of students reached the end of the interactive test more quickly.

	Reason Repetition / Reformulation	Rule Repetition / Reformulation	Immediate Rule Announcement
Small and medium scale companies	73%	7.6%	23%
Large scale company	64.7%	26.4%	32.3%
Graduate students	57.1%	10.7%	17.8%

Table 3 – The Percentages of Subjects within three Groups according to their Confirmation Tendencies

Our hypothesis for these two groups was that they would differ in terms of logical reasoning, and hence in their scores of the abstract questions found in pen-and-paper test. This difference was confirmed significantly in favor of the graduate students, when the number of subjects below and above median score were compared statistically ($\chi^2(1) = 8.114$, $p < .005$). Another interesting result was that the student group also performed better in questions with a software theme ($\chi^2(1) = 7.085$, $p < .01$).

Similar analyses were conducted among three more groups. All members who had software development and testing experience equal to or more than the average experience were included in the analyses. For the sake of simplicity, let us refer to the large scale company as C_1, the group of small and medium scaled companies as C_2 and graduate students as S. We first analyzed whether test results differed significantly among C_1 and C_2. Significant results were obtained with Mann-Whitney U-test for test severity, z = -2.48, p < .05 and eliminative/enumerative index, z = -3.82, p < .001. For test severity, the mean ranks were 23.36 for C_1 and 19.18 for C_2. For eliminative/enumerative index, the mean ranks were 27.07 for C_1 and 11.5 for C_2. These results indicated that, employees of the large scale company performed better in the interactive test compared to the employees of small and medium scaled companies when only subjects with an experience level higher than the average was taken into consideration.

Further analyses were conducted for the student group versus C_1 and C_2. It has been observed that the student group differed significantly from C_1 in terms of their scores in abstract questions ($\chi^2(1) = 6.773$, $p < .01$) and in software-domain questions ($\chi^2(1) = 7.38$, $p < .01$). A similar pattern of results was observed for the student group and C_2. Again, significant differences were found in the scores of the abstract questions ($\chi^2(1) = 6.677$, $p < .01$) and software-domain questions ($\chi^2(1) = 4.34$, $p < .05$). These results indicated that it was easier for the student group with the experience level equal to or more than the average experience to employ logical reasoning and use it effectively than software developers and testers of the companies.

The Analysis of Reasoning Skills. When we compared the strategies employed by the subjects in the pen-and-paper test, we observed that the percentage of falsifiers was higher in the group of students, while the percentage of verifiers and matchers was lower compared to the other group of software developers and testers of the four companies. This distribution is shown in Figure 4. The significant difference among the falsifiers, verifiers and matchers in both groups was confirmed statistically ($\chi^2(2) = 6.922$, $p < .05$).

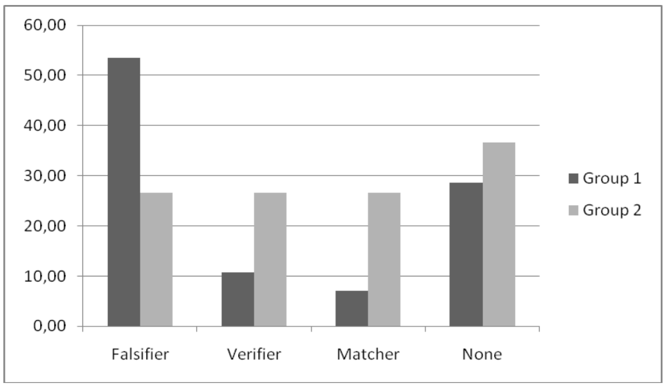

Figure 4 – The distribution of subjects according to Reich and Ruth's method among two groups, Group 1 being students and Group 2 being all employees

Since a high percentage of subjects could not be categorized according to this method, as an additional measure we have performed an analysis of 'insights'. Outcomes are shown in Figure 5, which seems to confirm the hypothesis that the percentage of people with complete insight is higher among students, whereas people with partial or no insight are less frequent in this group. These results were also confirmed statistically ($\chi^2(2) = 9.620$, $p < .01$). Hence, we can conclude that students performed better in pen-and-paper test when compared to all other subjects.

Further analyses were conducted to compare interactive test results. This time, S, the group of students, was compared to two groups, first being the large scale company C_1 and the second being the group of small and medium scaled companies C_2.

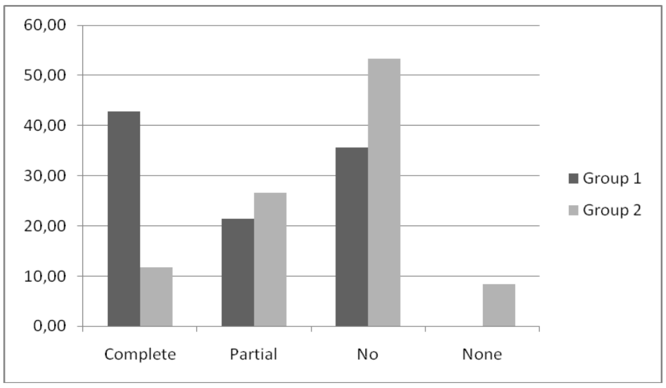

Figure 5 – The distribution of subjects according to insights among two groups, Group 1 being students and Group 2 being all employees

It has been observed that the time to finish the interactive test was significantly lower for S than C_1, $z = -3.57$, $p < .001$ and C_2, $z = -1.97$, $p < .05$. No significant difference was observed for eliminative/enumerative index.

When we compared test severities with Mann Whitney U-test, we have found a significant difference between S and C_1, $z = -2.46$, $p < .05$. S had an average rank of 24.72, while C_1 had an average rank of 35.99. This could mean that C_1 employed a better hypothesis testing strategy or exact the opposite, that they had strong confirmations. Looking again at Figure 3, we observed that the group of graduate students (G3 in the figure) also performed a desired hypothesis testing strategy starting with a mild test, increasing the severity and then again decreasing it. In order to be able to compare these two groups, we analyzed whether there was a significant difference between two groups in terms of confirmative behavior. In order to accomplish this, we took into account the subjects within groups who engaged in an activity such as repeating or reformulating a reason or a rule; or immediately announcing a rule. A significant difference was found between these groups in terms of the number of people engaging in a confirmative behavior ($\chi^2(2) = 5.939$, $p < .05$). We concluded that the group of graduate students was better at hypothesis testing. This was also supported by the fact that 27 out of 28 students found the correct rule at the end of the interactive test, whereas 29 out of 34 employees of the large scale company was able to find the rule. No significant difference was observed for test severity between S and C_2.

In order to observe more possible differences between the group of students and the large scale company, results of the pen-and-paper test are compared. Figure 6 shows the distribution of falsifiers, verifiers and matchers in both groups. This figure shows that the percentage of falsifiers among students is higher and the percentage of verifiers and matchers is lower.

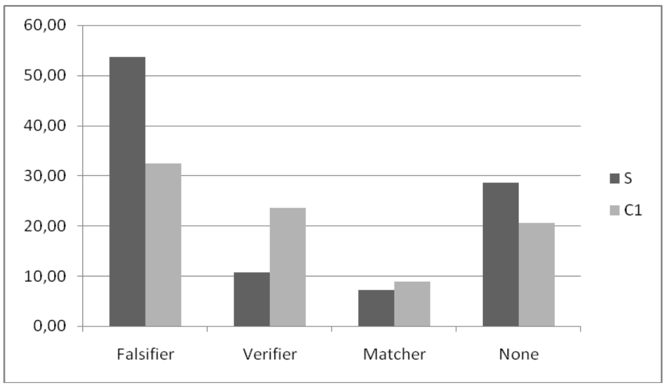

Figure 6 – The distribution of subjects according to Reich and Ruth's method among two groups, S being students and C1 being employees of the large scale company

It has been found that both groups differed significantly in terms of the percentages of falsifiers and verifiers, thus confirming the claim that there is a significant difference in both test performances between these groups ($\chi^2(1) = 4.835$, $p < .05$). The high percentage of subjects of the large scale company with no insight as shown in Figure 7 also confirms this fact.

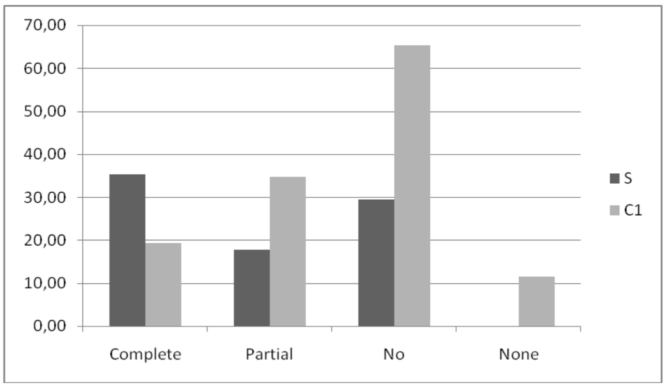

Figure 7 – The distribution of subjects according to insights among two groups, S being students and C1 being employees of the large scale company

5.2. Effects of Confirmation Bias on Software Defect Density

Analysis of the Effect of Confirmation Bias on Software Developer Performance. As a result of the lack of tendency to try to fail code during unit tests, a software developer is likely to introduce defects to his/her code. In order to analyze this phenomenon empirically, we analyzed the last ten releases of customer services and channel management software developed in the large scale telecommunications company C_1. There were 12 developers and 16 testers assigned to this software project. However, we could only perform our tests to five developers whose records appear in churn data. The rest of the development team were new to the project due to sudden change in the organizational structure.

During our analyses, we took into account only Java source codes, as churn data contained information about only Java source files. The files that are created in a release before the release of interest *R*, but just modified within release *R* are not taken into account. Our analyses include only Java source files that are created within each release. The owner of each file is determined from churn data and related defect information is obtained from the defect log of the corresponding release. As shown in Table 4, developer who gave up the interactive test (Developer1) has the highest defect ratio which is the ratio of the number of defected files to the total number of files implemented by that developer. Moreover, it took much longer for Developer1 to solve both parts of the pen-and-pencil test compared to the rest of the developers. On the other hand, no significant difference in elimination/enumeration index of Developer1 from the indices of the remaining developers was observed.

	Defect Ratio	Eliminative/ Enumerative Index	Interactive Test Duration (minutes)	Abstract & Thematic Test Duration (minutes)	Software Test Duration (minutes)
Developer1	0.86	0.83	ABORT	20	26
Developer2	0.38	1.83	12	12	10
Developer3	0.20	1.82	14	10	10
Developer4	0.00	0.75	22	9	13
Developer5	0.11	0.50	8	15	10

Table 4 – Defect Ratio versus Some Confirmation Bias Results of Developers of a Software Project of the Large Scale Telecommunications company (C_1).

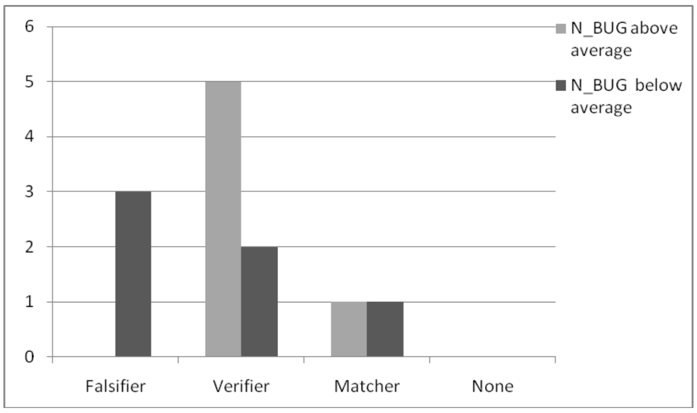

Figure 8- Distribution of falsifiers, verifiers, and matchers among testers who report bugs above and below average amount, according to Reich and Ruth's method.

Analysis of the Effect of Confirmation Bias on Software Tester Performance. In this part of our work, to analyze the effect of confirmation bias on tester performance, we inherited two tester performance metrics from tester competence reports of the large scale telecommunications company C_1. In this study we analyzed the testers of the same project group we performed the analyses about developer performance. Out of 16 testers, performance related data of 12 testers were in the tester competence reports. The remaining 4 testers had recently joined the project group due to the sudden change in the organizational structure. These metrics are the number of bugs reported (N_{BUG}) and the number of production defects caused (N_{PROD_DEF}) by each tester respectively. We grouped members of C_1 based on the values of N_{BUG} and N_{PROD_DEF}. Figure 8 shows that there are no falsifiers among testers who detect bugs above average N_{BUG} value. This group of testers also contains more verifiers compared to the tester group detecting bugs below average N_{BUG} value .

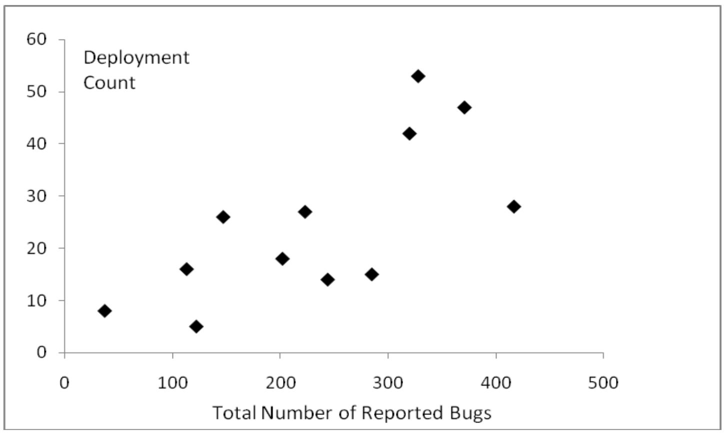

Figure 9- High Correlation between production defect and total number of reported bugs (spearman rank correlation: 0.8234)

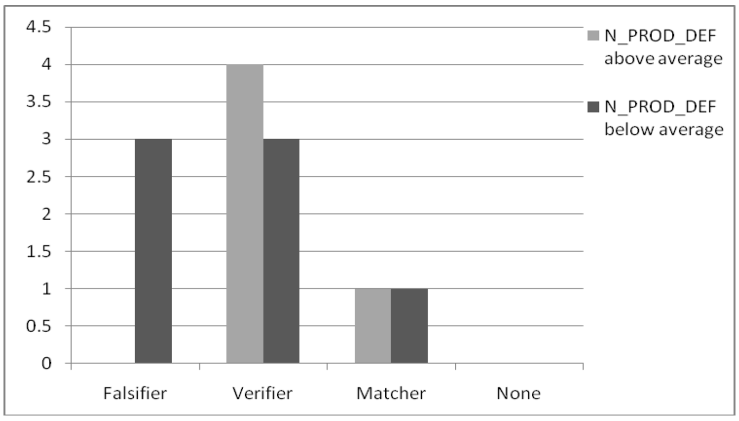

Figure 10- Distribution of falsifiers, verifiers, and matchers among testers who cause production defects above and below average amount, according to Reich and Ruth's method.

As shown in Figure 9, high correlation between total number of reported bugs and production defect count may indicate another phenomenon, namely, testers who report more bugs might be assigned codes with very high defect density requiring immense testing effort. However, for each tester there is also a time pressure to end the testing procedure and this may result in the deployment of the defected codes. Another explanation for the outcome shown in Figure 9, is that bugs are not classified according to their severities. Hence, large number of reported bugs does not necessarily mean that a significant portion of severe bugs has been reported. As a result, testers with low confirmation bias levels seem to detect more bugs. However they are more likely to overlook most of the defects which leads to an increase in production defects. In Figure 10, among testers who introduce production defects above average there are no falsifiers and this result is in line with our latter explanation.

7. Threats to Validity

In terms of internal validity, our quasi-independent variables are company size, experience and reasoning skills. In order to obtain measures for these variables, we performed both interactive and pen-and-paper tests to development and testing team working on a software project in the large scale telecommunication company (C_1) within a week. Tests were completed among graduate computer engineering students (S) also in less than a week; whereas the completion of the tests took 1 day for each of the small scale companies (C_2). Moreover, within any of the groups there was no event in between the confirmation bias tests that can affect subjects' performance.

However, problem may arise due to different experimental conditions. For instance, compared to graduate computer engineering students, stress factor of company workers due to the fact that they always have to rush the next release may have biased the results. In order to avoid mono-operation bias as a construct validity threat, we used more than a single dependent variable. We used Wason's elimination/enumeration index (Wason 1960), test severity in addition to interactive and pen-and-pencil test durations. As a result, we have avoided under-representing the construct and got rid of irrelevancies.

We have used three datasets to externally validate our results. We will continue expanding the size and variety of our dataset going forward. However, during our analysis to investigate the effect of confirmation bias on software defect density, we used data belonging to *only* five developers. This was partly because of the rapid and frequent changes in the development team. Out of 12 developers, only 5 of them were actively working on the project; while the rest were the newcomers and hence they have not started contributing to the project yet. We could not find the previous developers whose code commitment records were on the churn data, as most of them had left the company. In general, it is difficult to extract data that is informative about the defects introduced by a developer. Usually small and medium sized companies do not keep file-level defect information. Moreover, most companies do not classify defects according to their severity either. The data about the developers who contributed to a specific file, the dates of this contribution and defects related to the file differ from one company to another.

In order to statistically validate our results, we used Mann-Whitney U Test for continuous variables (e.g. test severity, eliminative/enumerative index, test durations). We used Mann-Whitney U Test, since we do not have any prior knowledge of the distribution of these values. For categorical variables such as number of falsifiers, verifiers, matchers, we used Chi Square test . Chi Square test was also used to evaluate the significance of the distribution of the subjects according to insights by Johnson-Laird and Wason.

8. Conclusion and Future Work

We have shown that there is no significant relationship between software development or testing experience and hypothesis testing skills. We concluded that experience did not play a role even in familiar situations such as problems about software domain.

The most striking difference was found between the group of graduate students and software developers and testers of the companies in terms of abstract reasoning skills. The fact that students scored better in software-domain questions although most of them had less software development and testing experience indicates that abstract reasoning plays an important role in solving everyday problems. It is highly probable that theoretical computer science courses have strengthened their reasoning skills and helped them to acquire an analytical and critical point of view. Hence, we can conclude that confirmation bias is most probably affected by continuous usage of abstract reasoning and critical thinking.

Company size was not a differentiating factor in abstract reasoning, but differences in hypotheses testing behavior was observed between two groups of companies grouped according to their sizes. The large scale company performed better in the interactive test, but it has been shown that the group of students outperformed this group in terms of both tests. This has led us to perform additional analyses and reach the conclusion that hypotheses testing skills were better in the group of students. Thus, we conclude that there is a relationship between confirmation bias and continuous usage of and training in logical reasoning and critical thinking.

The analysis we made among developers and testers of the large scale telecommunications company, showed that there is a direct correlation between confirmation bias and defect proneness of the code. This is due to the fact that including unit testing in the development phase, all levels of software testing should aim to fail the code, which implies that both testers and developers should have low confirmation bias levels. However, in order to obtain Statistically significant results we need more data.

As future work, we intend to increase the size of our data regarding total number of defects introduced by each developer per lines of code changes made by that developer. Recently, we are collecting data from a company developing software for baking services and this data shall belong to 100 software developers and testers. All these data will help us to empirically analyze the effect of confirmation bias on software defect density to obtain statistically significant results.

7. Acknowledgements

This research is supported in part by Turkish Scientific Research Council, TUBITAK, under grant number EEEAG108E014.

8. References

Calikli, G., Bener, A., and Arslan, B. (2010) An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers, to appear in the Proceedings of 32nd International Conference on Software Engineering (ICSE 2010).

Evans, J. St. B. T. (1972) Interpretation and matching bias in a reasoning task, Quarterly Journal of Experimental Psychology, 24, 193-199

Evans, J. St. B. T., Newstead, S.E., Byrne, R. M. J. (1993) Human Reasoning, The Psychology of Deduction.

Hilgard, E. R. (1938) A summary of alternative procedures for the construction of Vincent curves, Psychology Bulletin, 35, 282-297.

Kahneman, D., Slovic, P., and Tversky, A. (1982) Judgment Under Uncertainty: Heuristics and Biases, New York: Cambridge University Press

Poletiek, F. (2001) Hypothesis Testing Behavior (Essays in Cognitive Psychology), Psychology Press Ltd.

Reich, S.S. and Ruth, P. (1982) Wason's selection task: verification, falsification and matching. British Journal of Psychology, 73:3, 395-404.

Stacy, W. and MacMillan, J. (1993) Cognitive bias in software engineering, Communication of the ACM, 38, 6 (June 1995), 57-63.

Teasley, B., Leventhal, L. M., and Rohlman, S. (1993) Positive test bias in software engineering professionals: What is right and what’s wrong, In Empirical Studies of Programmers: Fifth Workshop

Wason, P. C. (1960) On the failure to eliminate hypotheses in a conceptual task, Quarterly Journal of Experimental Psychology (Psychology Press), 12. 129–140.

Wason, P. C. (1968) Reasoning about a rule, Quarterly Journal of Experimental Psychology (Psychology Press) 20: 273–28.