Empirical Software Engineering (2019) 24:1727–1761
https://doi.org/10.1007/s10664-018-9668-8

A controlled experiment on time pressure and confirmation bias in functional software testing

Iflaah Salman[1] · Burak Turhan[2] · Sira Vegas[3]

Published online: 18 December 2018

Abstract

Context Confirmation bias is a person's tendency to look for evidence that strengthens his/her prior beliefs rather than refutes them. Manifestation of confirmation bias in software testing may have adverse effects on software quality. Psychology research suggests that time pressure could trigger confirmation bias.
Objective In the software industry, this phenomenon may deteriorate software quality. In this study, we investigate whether testers manifest confirmation bias and how it is affected by time pressure in functional software testing.
Method We performed a controlled experiment with 42 graduate students to assess manifestation of confirmation bias in terms of the conformity of their designed test cases to the provided requirements specification. We employed a one factor with two treatments between-subjects experimental design.
Results We observed, overall, participants designed significantly more confirmatory test cases as compared to disconfirmatory ones, which is in line with previous research. However, we did not observe time pressure as an antecedent to an increased rate of confirmatory testing behaviour.
Conclusion People tend to design confirmatory test cases regardless of time pressure. For practice, we find it necessary that testers develop self-awareness of confirmation bias and counter its potential adverse effects with a disconfirmatory attitude. We recommend further replications to investigate the effect of time pressure as a potential contributor to the manifestation of confirmation bias.

Keywords Test quality · Software testing · Cognitive biases · Confirmation bias · Time pressure · Experiment · Human factors

Communicated by: Shin Yoo

✉ Iflaah Salman
iflaah.salman@oulu.fi

Extended author information available on the last page of the article.

1 Introduction

An important aspect of research related to human aspects in software engineering (SE) is human cognition. Human cognition is related to human thought processes. When a thought process is based on illogical reasoning, it might negatively impact the performed task (Calikli and Bener 2014). Formally, this phenomenon is known as cognitive bias and is defined as "*...cognitions or mental behaviours that prejudice decision quality in a significant number of decisions for a significant number of people*" (Arnott 2006, p. 59). Since the introduction of the concept by Tversky and Kahneman (1973), a considerable amount of work has been carried out to investigate cognitive biases in psychology (Gilovich et al. 2002).

Confirmation bias, a type of cognitive bias, is the tendency to look for confirmatory evidence rather than look for disconfirming information or evidence that refutes prior beliefs (Arnott 2006). The term *confirmation bias* was first used by P.C. Wason in his rule discovery experiment (Wason 1960). Confirmation bias belongs to the category of confidence biases, according to the taxonomy proposed by Arnott (2006). From Nickerson's point of view, confirmation bias is among the top reasons for problematic human reasoning (Nickerson 1998). Multiple psychological studies have investigated and confirmed the presence of confirmation bias attributed to multiple antecedents and how substantial and strong their effects are (Nickerson 1998; Ask and Granhag 2007; Hernandez and Preston 2013). For example, psychological studies have explored and documented time pressure as an antecedent to the manifestation of confirmation bias, e.g. time pressure caused criminal investigators and jurors to stick to their initial beliefs while evaluating witness' testimonies (Ask and Granhag 2007) and deciding on the verdict (Hernandez and Preston 2013) respectively, which affected their evaluations. Time pressure in the software organisational context is a substantial and unavoidable phenomenon that is usually perceived to have a negative influence, e.g. it causes developers to take shortcuts (Mäntylä et al. 2014). Studies in SE investigating the effects of time pressure report both positive and negative effects, e.g. time pressure was identified to demotivate the process of software improvement (Baddoo and Hall 2003), suspected to deteriorate quality in the development cycle and global software testing (Wilson and Hall 1998; Shah et al. 2014), and improve the efficiency of requirements reviews and test case development (Mäntylä et al. 2014).

Software engineering is also subject to the effects of cognitive biases, especially to those of confirmation bias (Stacy and MacMillan 1995). In this paper, we limit the scope of the investigation to functional software testing. The manifestation of confirmation bias among software testers is considered based on their inclination towards designing test cases that confirm the correct functioning of an application rather than scenarios that break the code (Calikli and Bener 2014; Teasley et al. 1994; Salman 2016). In this regard, limited empirical evidence is available in the SE literature that reports confirmation bias manifestation in software testing resulting in deteriorated software quality, e.g. higher defect rate (Calikli and Bener 2010) and an increased number of production defects (Calikli et al. 2013).

Despite the recognition of time pressure and confirmation bias' effects, a research gap exists for studying time pressure as an antecedent to confirmation bias in software testing. It is a combination that may exacerbate software quality and deserves empirical attention per evidence available from the psychology discipline (Ask and Granhag 2007; Hernandez and Preston 2013). According to Mäntylä et al. (2014) and Kuutila et al. (2017), the work on the effects of time pressure in software testing is still scarce and requires further investigation. In addition to this, the motivation for studying time pressure as a potential antecedent to cognitive biases is based on the results of our systematic mapping study (SMS) on the topic of cognitive biases in SE (Mohanani et al. 2018).

Therefore, our goal is *to examine the manifestation of confirmatory behaviour by software testers and the role of time pressure in its promotion*. Specifically, the purpose of this study is *to investigate whether testers exhibit confirmatory behaviour when designing functional test cases and how it is impacted by time pressure*. We achieve this goal by conducting a controlled experiment with graduate students where we observe confirmation bias in designing functional test cases in terms of their consistency and inconsistency with the requirements specification provided. The design of our experiment is one factor with two levels, i.e. time pressure vs no time pressure. Our results confirm previous research findings: participants designed considerably more confirmatory test cases than disconfirmatory ones. We also observed that time pressure did not increase the rate of confirmatory behaviour in the group working under time pressure.

This study enhances the body of SE knowledge through the following contributions:

– An experimental design studying the effects of time pressure on confirmation bias, a novelty in SE within the context of software testing.
– A new perspective when compared to earlier work in defining the (dis)confirmatory nature of tests, i.e. *consistent* (c) and *inconsistent* (ic) test cases. The change in terminology and the need for it are elaborated in Section 3.2.
– Empirical evidence in support of confirmatory behaviour in functional test case design.
– Empirical evidence suggesting that time pressure may not increase the rate of confirmatory behaviour in functional test case design.

Section 2 presents the related work on time pressure and confirmation bias. In Section 3 the research method and experiment planning are explained in detail, which is followed by details regarding the experiment execution in Section 4. The results are presented in Section 5 and discussed in Section 6. In Section 7, we describe the threats to validity. Finally, Section 8 concludes our work and discusses possible future extensions.

2 Related Work

In this section, we present a brief overview of the SE and psychology literature on confirmation bias, its detection/measurement, and time pressure.

2.1 Confirmation Bias in Software Testing

The studies reviewed in this section were found as a result of our SMS (Mohanani et al. 2018). Most work on cognitive biases in the context of software quality and testing is focused on confirmation bias (Mohanani et al. 2018; Salman 2016). Another term related to confirmation bias is *positive test bias* or *positive test strategy*, found in the works of Teasley et al. (1994) and Leventhal et al. (1994). Leventhal et al. (1994) refer to the work of Klayman and Ha (1989) and mention that the phenomenon of *positive test bias* is also called *confirmation bias*. According to the definition of confirmation bias, the more the bias exists among the testers the more it negatively affects the testing (Calikli and Bener 2013, 2014; Leventhal et al. 1994).

In 1994, Teasley et al. conducted an experimental study to investigate the effects of positive test strategy on functional software testing. They also investigated the impacts of expertise level and the level of detail of specifications on positive test strategy. They found that the use of a positive test strategy decreased with higher expertise level, but the influence of the completeness of the specifications remained inconclusive because only one experiment showed support for it. In another study by the same authors Leventhal et al. (1994), an additional factor (error feedback), along with the previous factors, was studied with the same objective. The results of the study revealed that complete specifications, as well as higher expertise, may aid in mitigating positive test bias. The effect of error feedback, however, remained inconclusive, which the authors linked to the types of software used in the studies (Leventhal et al. 1994). In this family of experiments, Teasley et al. (1994) and Leventhal et al. (1994) recruited senior level and graduate students in computer science to represent advanced testers.

With the aim of finding evidence for positive test bias, Causevic et al. (2013) conducted a test-driven development (TDD) experiment in industry. The authors found a significant difference between the number of positive and negative test cases developed by the participants. In addition, negative test cases were found to have a greater possibility to detect defects compared to positive test cases. The authors measured the quality of the test cases by the quality of code for each test case and observed differences in the quality of the negative and positive test cases (Causevic et al. 2013).

In recent years, multiple studies have been conducted by Calikli and Bener to explore several factors that might affect confirmation bias (Calikli and Bener 2010, 2014; Calikli et al. 2010a, b). For instance, in a preliminary analysis, the authors studied company culture, education and experience as factors and found that only company culture affected confirmation bias levels (Calikli et al. 2010b). Later, Calikli and Bener (2010) studied the effects of experience and activeness (in development and testing), along with the logical reasoning skills gained through education, on confirmation bias. In 2014, they extended their set of factors to include job title (developer, tester, analyst, and researcher), education, development methods and company size (small and medium enterprises [SME] vs large-scale companies) (Calikli and Bener 2014). Both studies showed no effect of either development methods or experience in development and testing on confirmation bias, but as an effect of logical reasoning skills, low bias levels were observed (Calikli and Bener 2010, 2014). Low confirmation bias levels were observed in participants who were experienced but inactive in development or testing (Calikli and Bener 2010). No effect of company size, educational background (undergraduate) or educational level (bachelor's, master's) were observed (Calikli and Bener 2014). However, the job title (researcher) was associated with significantly lower levels of confirmation bias when compared to the rest of the job titles, which is possibly due to researchers' acquired critical and analytical skills (Calikli and Bener 2014).

2.2 Measurement of Confirmation Bias

The first approach identified in the software testing literature uses test artefacts to measure the manifestation of confirmation bias (Teasley et al. 1994; Leventhal et al. 1994). These studies measured *positive test bias* by mapping it onto functional software testing as: *if testing is done with the data provided in the specifications, then testing is done in a hypothesis-consistent way, but if testing is done with the data outside the specifications then it is in hypothesis-inconsistent way*. Leventhal et al. (1994) used equivalence classes and boundary conditions to classify test cases as positive or negative. According to Leventhal et al. (1994), positive test cases deal with valid equivalence classes and negative test cases with invalid equivalence classes. Causevic et al. (2013), on the other hand, declared a test case to be negative if it exercised a program in a way that was not explicitly declared in the requirements specifications and the test cases testing the implicitly forbidden behaviour of the program were also considered negative.

The second approach is explained in detail by Calikli et al. (2010a), who derived their measurements from psychological instruments based on the work of Wason (1960). Rather than measuring any testing artefact produced by the participants, the authors used Wason's Rule Discovery and Selection Task, after some modifications, to assess how people think. For Wason's Selection Task, the questions that were related to the software domain required an analysis of a software problem that was independent of programming tools and environment. (Calikli and Bener 2010; Calikli et al. 2010b)

2.3 Time Pressure

This section focuses on the factor of time pressure from two perspectives. First, we present the SE studies that have considered time pressure in different SE contexts. Then, we shift the focus to the psychology studies discipline that motivate our study.

2.3.1 Software Engineering

Table 1 presents the studies that examined or observed time pressure (TP) in different contexts of SE. All the studies listed in the table are selected from two recent studies by Mäntylä et al. (2014) and Kuutila et al. (2017) based on the criterion that a study belongs to the SE domain.

The first column in Table 1 names the study and lists its type, i.e. either qualitative or quantitative. The second column mentions the SE context and the objective of the study, whereas the third column presents the results of the respective study from the perspective of the time pressure factor. In the table, five out of twelve studies are related to software quality and software testing and four are related to software development. Three studies in Table 1 reported that time pressure did not have a negative effect, e.g. impact, on task performance (Topi et al. 2005). The studies in Table 1 also report the positive effects of time pressure, e.g. improved effectiveness of the participants when not working as individual testers (Mäntylä and Itkonen 2013). Finally, time pressure has been found to be a negative factor in the investigated contexts by Baddoo and Hall (2003) and Wilson and Hall (1998).

2.3.2 Psychology

This section elaborates on confirmation bias as a psychological phenomenon and on how psychology literature has operationalised time pressure to study its influence on confirmatory and motivational behaviours. The studies in this section report time pressure as a factor that negatively impacted the process/quality by influencing the manifestation of confirmation bias in the studied contexts. Of particular importance to our work, Ask and Granhag (2007) reported that the tendency of people to rely on prior beliefs rather than on external information increased under time pressure. Hence, we aim to take an interdisciplinary perspective.

Hulland and Kleinmuntz (1994) experimentally examined if greater time pressure causes decision-makers to rely on the summary evaluations retrieved from memory as aopposed to relying on information from the external environment as a way to avoid effort expenditure. The authors observed changes in the search and evaluation process but they did not observe greater reliance on summary evaluations. Hernandez and Preston (2013) carried out a study in the context of juror decision making that examined the effect of difficulty in processing available information (disfluency) with confirmation bias. They paired disfluency with other cognitive loads, including time pressure, and found that with the presence of both

Table 1 Time pressure (TP) studies in SE in chronological order

Study & Type	Context & Aim	Results
Wilson and Hall (1998); Qualitative	Software quality; investigation of the software engineer's views on quality.	TP is suspected to have an overall negative impact on the development cycle.
Austin (2001); Theoretical	Software quality; derives a relationship between deadline setting policies and product quality.	Setting aggressive deadlines can eliminate taking shortcuts.
Baddoo and Hall (2003); Qualitative	Software process; issues demotivating the software practitioners for software process improvement (SPI).	TP identified as one of the demotivating factors for SPI.
Topi et al. (2005); Quantitative	Software development (database query); investigation of the relation between time availability and task complexity.	Time availability did not affect task performance. However, task complexity influenced performance at all levels of time availability.
Hazzan et al. (2007); Qualitative	Software engineering; time management perspective in SE.	Time tracking of software projects through a tight development process (agile) because it is difficult to track the time dimension.
Harris et al. (2009); Theoretical	Software development; the role of time and TP in design and performance of agile processes.	Proposition: employees motivation and stress is higher when deadlines are shorter.
Nan and Harter (2009); Qualitative	Software development; effects of schedule pressure and budget on software cycle time and effort.	Development outcomes did not have a significant nonlinear impact of schedule pressure.
Cataldo (2010); Qualitative	Distributed development; implications for collaborative tools after analysing the sources of errors in distributed development projects.	TP is found to be a relevant source of error in distributed development projects.
Mäntylä and Itkonen (2013); Quantitative	Software testing; effect of multiple individuals working on the same task and TP on the effectiveness and efficiency of manual testing.	Multiple time-pressured testers delivered better defect detection compared to individual testers under no time pressure.
Shah et al. (2014); Qualitative	Global software testing (GST); perception of testing and deadline pressure.	TP is perceived negatively by the test engineers in terms of quality. However, team configurations in GST do affect the perception and experience of TP.
Mäntylä et al. (2014); Quantitative	Software testing; effects of TP on effectiveness and efficiency. Effect of knowledge on TP and the perception of TP.	TP did not cause negative effects on effectiveness. No effect of knowledge was observed on TP. TP improved the efficiency in requirements review and test case development.

Table 1 (continued)

Study & Type	Context & Aim	Results
Kuutila et al. (2017); Qualitative	Software engineering; review of the literature on TP in SE and related professions.	Proposed a list of testable hypotheses for studying the effects of TP on developers' mental health.

disfluency and time pressure, confirmation bias could not be overcome. The authors applied time pressure by restricting the submission of the verdict within a prespecified time frame. Another study was performed in the same context (investigative psychology) and studied the behaviour of criminal investigators. They evaluated witness' testimonies as either confirming or disconfirming the central hypothesis (Ask and Granhag 2007). The results of the study revealed that participants under high time pressure were more likely to stick with their initial beliefs and were less affected by the subsequent evidence. The authors created high time pressure by orally informing the participants that they had a limited amount of time to complete the task. In addition to this oral instruction they were encouraged to complete the task faster and were informed when there was five minutes remaining.

The manifestation of confirmation bias under time pressure in investigative work can be related to its manifestation in functional software testing. For example, a juror reading a case description, where the defendant's case is objectively described, is more likely to reach a guilty verdict when the same juror previously read a psychologist's testimony highlighting the negative aspects of the defendant's personality profile (Hernandez and Preston 2013). In this situation, the juror manifests confirmation bias in reaching the guilty verdict, which is tentatively formed (e.g. biased upon) after reading the psychologist's testimony. Similarly, a software tester, *preconditioned* on what is provided in the specifications, may manifest a *specification-confirming* behaviour in testing. The tester, under time pressure, is more likely to follow the confirming attitude and may not necessarily consider, or give priority to, testing the other scenarios that may occur (e.g. corner cases) but are not specified in the requirements.

2.4 Research Gap

Previous work in the SE domain is limited to investigations of the effects of time pressure from productivity and quality perspectives. In other words, none of the past studies explored the impacts of time pressure on the artefacts produced by software engineers as a consequence of a certain cognitive bias; thus, they have neglected the human factors. Additionally, the studies that have been performed that consider the human aspects of software testing, particularly cognitive biases, lack the investigation of the time pressure factor as an antecedent. In comparison to the presented psychological studies, investigating time pressure's effect on confirmation bias in an engineering discipline differentiates our study.

Consequently, to bridge the gap, we contemplate the manifestation of confirmatory behaviour along with the investigation of time pressure's effect on the manifestation of confirmation bias in the artefacts prepared by software testers. Instead of measuring confirmation bias as found in the existing literature, our approach is to detect it from the produced artefacts (explained in Section 3.2). Furthermore, we use experimentation, which is method-wise similar to the research approaches taken by, e.g. Hernandez and Preston (2013), Mäntylä et al. (2014) and Teasley et al. (1994). To summarise, our study is novel in that it investigates the effect of time pressure on confirmation bias in software testing.

3 Research Method

In this section, we present the details of our experiment from the perspective of definition and planning - the first two stages of the experiment process (Wohlin et al. 2000). To enable further replications, we share the experimental protocol and scripts at: https://doi.org/10.5281/zenodo.1193955.

3.1 Goal

Our objective is to examine whether testers manifest confirmation bias (leading to a confirmatory attitude) during testing and whether time pressure promotes the manifestation of confirmation bias. The aim of our research, according to Goal-Question-Metric (Basili et al. 1994) is as follows:
Analyse the functional test cases
For the purpose of examining the effects of time pressure
With respect to confirmation bias
From the point of view of researchers
In the context of an experiment run with graduate students (as proxies for novice professionals) in an academic setting.

Consequently, the research questions of our study are:

RQ1: Do testers exhibit confirmatory behaviour when designing functional test cases?

RQ2: How does time pressure impact the confirmatory behaviour of testers when designing functional test cases?

3.1.1 Context

We study the manifestation of confirmation bias in functional software testing and examine how time pressure promotes confirmation bias manifestation in the same context. The phenomenon is studied in the context of a controlled experiment in academic settings with first-year master's degree students, enrolled in the Software Quality and Testing course at the University of Oulu, as proxies for novice professionals. We limit our investigation to functional (black box) testing, which was part of the curriculum of the aforementioned course. For the purposes of the experiment, we focus only on the design of functional test cases, not their execution. We aim for an implementation-independent investigation of the phenomenon, since we are interested in studying the mental approach of the study participants in designing test cases, which precedes their execution. Besides, in system, integration and acceptance testing, software testers (by job role/title) design test cases before the code actually exists, using the requirements specifications. Therefore, our scope is limited to determining the type (i.e. consistent or inconsistent with the specifications) of functional tests designed by the participants, rather than their execution or fault-detection performance. We use a realistic object for the task of designing functional test cases under time pressure and no time pressure conditions.

3.2 Variables

This section elaborates on the independent and dependent variables of our study.

3.2.1 Independent Variable

The independent variable of our study is **time pressure** with two levels: time pressure (TP) and no time pressure (NTP). To decide on the duration for the two levels, we executed a pilot run (explained in detail in Section 3.8) with five postgraduate students. It took 45 min, on average, for the pilot participants to complete the task. Accordingly, we decided to allocate 30 min for the TP group and 60 min for the NTP group to operationalise time pressure.

The timing of the task was announced differently to the experimental groups. The experimenter reminded the participants in the TP group thrice of the remaining time; the first reminder was after fifteen minutes had elapsed and the rest of the reminders were given every five minutes thereafter. This was done to psychologically build time pressure. In contrast, after the initial announcement of the given duration to the NTP group, no further time reminders were made. This is in line with how Hernandez and Preston (2013) and Ask and Granhag (2007) operationalised time pressure in their studies.

3.2.2 Dependent Variables

Our study includes three dependent variables, which are **c** - number of consistent test cases, **ic** - number of inconsistent test cases and **temporal demand**. We define these dependent variables as follows:

Consistent test case: A consistent test case tests strictly according to what has been specified in the requirements, i.e. consistency with the specified behaviour. In the context of testing this refers to: 1) the defined program behaviour on a certain input; and 2) the defined behaviour for a specified invalid input. **Example:** If the specifications state, *"... the phone number field does not accept alphabetic characters..."*, the test case designed to validate that phone number field does not accept alphabetic characters is considered a consistent test case.

Inconsistent test case: An inconsistent test case tests the scenario or the data input that is not explicitly specified in the requirements. We also consider such test cases that present outside-of-the-box thinking at the tester's end inconsistent. **Example:** If the specifications only state, *"... the phone number field accepts digits..."*, and the application's behaviour for the other types of input for that field is not specified, then the following test case is considered inconsistent: *the phone number field accepts only the + sign from the set of special characters (e.g. to set an international call prefix).*

In contrast to Leventhal et al. (1994), we do not consider a test case validating an input from an invalid equivalence class as inconsistent, as long as it is specified in the requirements. On the contrary, we consider it consistent, because the tester has exhibited a confirmatory behaviour by conforming to what s/he has been informed to validate. If test cases are classified using our *consistent* and *inconsistent* definitions and Causevic et al.'s (2013) positive and negative definitions, the results might be the same. However, the outside-of-the-box thinking is an additional aspect of our definition of inconsistent, which considers the completeness of the requirements specification in the light of the context/domain. Unlike Calikli et al. (2010a), we do not utilise any tests from psychology to measure confirmation bias. Instead, we detect its manifestation by analysing the test artefacts of the participants and do not directly observe how people think or what their thinking inclinations, in general, are.

We therefore introduce the terms *consistent* and *inconsistent* in order to distinguish the concept from previous forms of measurement of confirmation bias, based on the contradictory understandings of a positive and negative test case. For example, Leventhal et al. (1994) and Causevic et al. (2013) use the same terminology but measure confirmation bias differently. We believe that our proposed terminology is more straightforward to comprehend as compared to the potential ambiguity of positive/negative terminology.

Temporal demand: We use the NASA task load index (TLX) as a measure of task difficulty as perceived by the participants. We apply the same definition of temporal demand as defined in the NASA-TLX, which is the degree of time pressure felt due to the pace or tempo at which events take place (Research Group HP and Ames Research Center N 1987). Therefore, it captures the time pressure perceived by the participants in the experimental groups.

3.3 Data Extraction and Metrics

The section elaborates on the data extraction and metrics defined for capturing confirmation bias and temporal demand.

3.3.1 Proxy Measure of Confirmation Bias

We mark the functional test cases designed by the participants as either consistent (c) or inconsistent (ic). To detect the bias of participants through a proxy measure, we derive a scalar parameter based on (c) and (ic) test cases designed by the participants and the total count of (all possible) consistent (C) and inconsistent (IC) test cases for the given specification:

$$z = c/C - ic/IC$$

- if $z > 0$; participant has designed relatively more consistent test cases
- if $z < 0$; participant has designed relatively more inconsistent test cases
- if $z = 0$; participant has designed a relatively equal number of consistent and inconsistent test cases.

The value of z is the rate of change in relative terms. It is the difference of consistent test case coverage and inconsistent test case coverage. It indicates one of the above three conditions within the range $[-1, +1]$. In terms of confirmation bias detection, $z = 0$ means the absence of confirmation bias. If z is $+1$, then it indicates the maximum manifestation of confirmation bias, because only consistent test cases are designed with complete coverage, and -1 is an indication of designing inconsistent test cases only with complete coverage. We should note that although -1 is an unusual case, indicating that no consistent test cases were designed at all, it depicts a situation in which no bias has manifested. This case is quite impractical to occur because it suggests that no test case validating the specified behaviour of the application was designed. For the purposes of measurement, we predesigned a complete set of test cases (consistent and inconsistent) based on our expertise from a researcher's perspective. Our designed set comprised 18 consistent (C) and 37 inconsistent (IC) test cases. The total number of (in)consistent test cases are defined in absolute numbers in order to be able to compare and perform an analysis in relation to a (heuristic) baseline. In order to enhance the validity of the measures, we extended the set of our predesigned test cases after the experiment with the valid test cases (consistent and inconsistent) designed by the participants that were missing from our set. This improvement step is in line with Mäntylä et al. (2014), who mentioned that it helps to improve the validity of the results. As a result, our test set includes 18 (C) and 50 (IC) test cases in total.

3.3.2 Temporal Demand

In order to capture temporal demand (TD), we used the values of the rating scale marked by the participants on NASA-TLX sheets. The scale ranges from 0 to 100, i.e., from low to high temporal demand perceived for the task (Research Group HP and Ames Research Center N 1987).

3.4 Hypothesis Formulation

According to the goals of our study, we formulate the following hypotheses:

H1 states that: *Testers design more consistent test cases than inconsistent test cases.*

$$H1_A : \mu(c) > \mu(ic)$$

and the corresponding null hypothesis is:

$$H1_0 : \mu(c) \leq \mu(ic)$$

H1's directional nature is attributed to the findings of Teasley et al. (1994) and Causevic et al. (2013). Their experiments revealed the presence of *positive test bias* in software testing.

As an effect of time pressure on consistent and inconsistent test cases, our second hypothesis postulates the following:

H2: (Dis)confirmatory behaviour in software testing differs between testers under time pressure and under no time pressure.

$$H2_A : \mu([c, ic]_{TP}) \neq \mu([c, ic]_{NTP})$$

$$H2_0 : \mu([c, ic]_{TP}) = \mu([c, ic]_{NTP})$$

While *H2* makes a comparison in absolute terms, the third hypothesis considers the effect of time pressure on confirmation bias in relative terms with the given coverage to the specifications (z). Accordingly *H3* states:

H3: Testers under time pressure manifest relatively more confirmation bias than testers under no time pressure.

$$H3_A : \mu(z_{TP}) > \mu(z_{NTP})$$

$$H3_0 : \mu(z_{TP}) \leq \mu(z_{NTP})$$

The directional nature of *H3* is based on the evidence from the psychology literature in which time pressure was observed to increase confirmation bias in the studied contexts (Ask and Granhag 2007; Hernandez and Preston 2013).

To validate the manipulation of the levels of the independent variable - time pressure vs no time pressure - we formulate a posthoc sanity check hypothesis that postulates:

H4: Testers under time pressure experience more temporal demand than testers under no time pressure.

$$H4_A : \mu(TD_{TP}) > \mu(TD_{NTP})$$

$$H4_0 : \mu(TD_{TP}) \leq \mu(TD_{NTP})$$

3.5 Design

We chose a one factor with two levels between-subjects experimental design for its simplicity and adequacy to investigate the phenomenon of interest, as opposed to alternative designs. Table 2 shows the design of the experiment, where ES stands for experimental session and TP and NTP stand for the time pressure and no time pressure groups, respectively. In addition, this design is preferable as it does not introduce a confounding factor for task-treatment interaction, thus it enables the investigation of the effects of the treatment and control on the same object in parallel running sessions.

Table 2 Experimental design

	TP	NTP
ES	Group 1	Group 2
Object (Task)	MusicFone	MusicFone

3.6 Participants

We employed convenience sampling in order to draw from the target population. The participants were first-year graduate-level (master's) students registered in the Software Quality and Testing course offered as part of an international graduate degree programme at the University of Oulu, Finland, in 2015. The students provided written consent for the inclusion of their data in the experiment. All students were offered this exercise as a non-graded class activity, regardless of their consent. However, we encouraged the students to participate in the experiment by offering them bonus marks as an incentive. This incentive was announced in the introductory lecture of the course at the beginning of the term. In the data reduction step, we dropped the data of those who did not consent to participate in the experiment. This resulted in a total of 43 experimental participants.

Figure 1 presents the clustered bar chart showing the academic and industrial experience of 43 participants in software development and testing. Along the $y - axis$ are the percentages depicting the participants' range of experience in the categories presented along the $x - axis$. The experience categories are: Academic Development Experience (ADE), Academic Testing Experience (ATE), Industrial Development Experience (IDE) and Industrial Testing Experience (ITE). The four experience range categories are less than 6 months, between 6 months and one year, between 1 and 3 years and more than 3 years.

More than 80% of the participants have less than 6 months of testing experience both in academia and industry, which is equivalent to almost no testing experience. This indicates

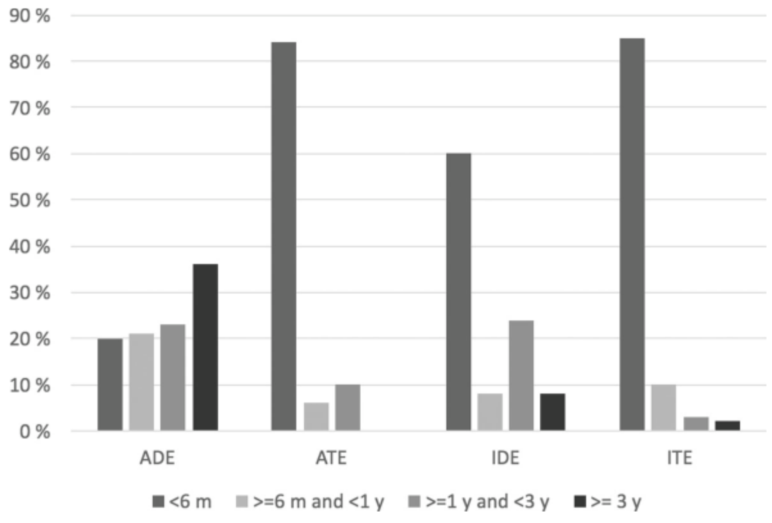


Fig. 1 Experience of participants

Table 3 Training sequence

Session	Lecture I	Class exercise	Lecture II
Duration	2 hours	2 hours	2 hours
Content	Functional testing techniques. (equivalence partitioning, boundary value analysis)	Functional test case design using the provided template; solution discussion.	Test case design techniques (need and classification of techniques, control flow techniques, functional techniques); test case specification.

that our participants have much less experience in testing when compared to development. The second highest percentages of experience are in the $>= 1y$ and $< 3y$ range, except for in ITE. The pre-questionnaire data also shows that 40% of the participants have industrial experience, i.e. more than 6 months. Thirty-two percent marked their industrial experience in development and testing based on the developer or tester roles rather than considering testing as part of a development activity.

Considering our participants' experience and the degree in which they are enrolled, we can categorise them as proxies for novice professionals according to Salman et al. (2015).

3.6.1 Training

Before the actual experimental session began, the students enrolled in the course were trained on functional software testing, as part of their curriculum, in multiple sessions. They were taught about functional (black box) testing over two lectures. In addition, one lecture was reserved for an in-class exercise where the students were trained for the experimental session using the same material but with a different object (requirements specification) to gain familiarity with the setup. Specifically, the in-class exercise consisted of designing functional test cases from a requirements specification document using the test case design template that was later used in the actual experiment as well. However, different from the actual experimental session, students were not provided with any kind of requirements supporting screenshot, which was available during the experimental sessions. One of the authors discussed students' test cases in the same training session for giving feedback. In every lecture, we specifically taught and encouraged the students to think up inconsistent test cases. Table 3 shows the sequence of these lectures with the major content related to the experiment. We sequenced and scheduled the lectures and the in-class exercise to facilitate the experiment.

3.7 Experimental Materials

The instrumentation that we developed to conduct the experiment consisted of a pre-questionnaire, the requirements specification document with a supporting screenshot, the test case design template and a post-questionnaire. Filling in the test case design template and post-questionnaire are both pen-and-paper activities, whereas the pre-questionnaire was administered online. The experimental package consisting of all the above mentioned materials is available from this URL.[1]

[1]https://doi.org/10.5281/zenodo.1193955

3.7.1 Pre/Post-questionnaires

Having background information on the participants aids in their characterisation. The pre-questionnaire was designed using an online utility and collected information regarding participants' gender, educational background, and academic and industrial development and testing experience. Academic experience concerned collecting information regarding the development and testing performed by the participants as part of their degree programme courses on development and testing. Industrial experience questions collected information on the testing and development performed by the participants in different roles (as a developer, tester, designer, manager and other).

We used hardcopy NASA-TLX scales to collect post-questionnaire data for two reasons. The first reason is that it is a well-known instrument for measuring task difficulty as perceived by the task performer (Mäntylä et al. 2014). Second, one of the load attributes with which the task difficulty is measured, i.e. temporal demand, captures the time pressure experienced by the person performing the task. In this respect, it was useful to adopt NASA-TLX because it also aided us in assessing (i.e. $H4$) how well we administered and manipulated our independent variable, i.e. time pressure in terms of the temporal demand felt by the participants.

3.7.2 Experimental Object

We used a single object appropriate for the experimental design. The task of the participants was to design test cases for the requirements specification document of the experimental object, i.e. MusicFone, that has been used in many reported experiments as a means of simulating a realistic programming task, e.g. by Fucci et al. (2015). MusicFone is a GPS-based music playing application. It generates a list of recommendations of artists based on the artist currently being played. It also displays the artists' upcoming concerts and allows for the planning of an itinerary for the user depending upon their selection of artists and based on the user's GPS location. Hence, choosing MusicFone as an object is an attempt to mitigate one of the non-realism factors (non-realistic task) of the experiment, as suggested by Sjoberg et al. (2002).

MusicFone's requirements specification document is originally intended for a lengthy programming-oriented experiment (Fucci et al. 2015). In order to address our experimental goals and to abide by the available experiment execution time, we modified the requirements specification document so that it could be used for designing test cases within the available time frame. Leventhal et al. (1994) defined three levels of specifications: minimal (features are sketched only), positive only (program actions on valid inputs) and positive/negative (program actions on valid and any invalid input) specifications. If we relate the completeness of our object's (MusicFone) specifications to the Leventhal et al.'s (1994), it is closer to a positive only specification document. We cannot classify MusicFone into the third category because the specifications stated the required behaviours but contained less information on handling the non-required behaviour of the application, thus qualifying our object as having a realistic level of specifications (Davis et al. 1993; Albayrak et al. 2009).

In addition to the requirements specification document, we also provided a screenshot of the working version of the MusicFone application UI to serve as a conceptual prototype to enable a better understanding of the application. We ensured that the screenshot of the developed UI was consistent with the provided requirements specification because the presence of errors or feedback from the errors could possibly affect the testing behaviour in terms of a positive test bias (Teasley et al. 1994). In other words, if a tester finds an error then s/he might possibly start looking for more similar errors, which will impact the behaviour of the testing and leads to negative testing (Teasley et al. 1994). However, the participants did not interact with the UI in our experimental setting, as we considered interaction as part of the execution of tests rather than their design.

3.7.3 Test Case Design Template

To ensure consistency in the data collection, we prepared and provided a test case design template to the participants, as shown in Table 4. This template consists of three main columns: test case description, input/pre-condition and expected output/post-condition, along with an example test case. The example test case is provided so that the participants know the level of detail required when designing test cases. Furthermore, these three columns were chosen to aid us in understanding the designed test cases better during the data extraction phase, e.g. in marking them as consistent or inconsistent.

3.8 Pilot Run

We executed a pilot run with five postgraduate students to achieve two objectives. First, was to decide on the duration of the two levels (TP, NTP), and second was to improve the instrumentation of our experiment. In addition to meeting the first objective, we improved the wording of the requirements specification document to increase comprehension, based on the feedback from the participants. Two authors of this study independently marked the pilot run test cases as (in)consistent and then resolved discrepancies via discussion. This ensured they shared a common understanding and this knowledge was then applied by the first author in the data extraction.

3.9 Analysis Methods

We analysed the data by preparing descriptive statistics followed by the execution of statistical significance tests of the *t-test* family and *F-test* (Hotelling's T^2) to test our hypotheses. To properly perform the statistical tests, we checked whether the data met the chosen test's assumptions; in the event that it failed to meet assumptions, a non-parametric counterpart of the respective test was performed. Hotelling's T^2 assumes the following:

1. There are no distinct subpopulations and populations of samples have unique means.
2. The subjects from both populations are independently sampled.
3. The data from both populations have a common variance-covariance matrix.
4. Both populations have a multivariate normal distribution.

For univariate and multivariate normality assumptions, the Shapiro-Wilk test was used. We report multiple types of effect sizes depending upon the statistical test run and assumptions considered by the respective effect size measure (Fritz et al. 2012). Cohen's d (0.2 = small, 0.5 = medium, 0.8 = large) and correlation coefficient r (0.10 = small, 0.30 = medium, 0.50

Table 4 Test case design template

ID	Description	Input/Pre-condition	Expected Output/Post-condition
1	Application displays 20 artists to the AppUser.	Get Recommendations clicked; artists from Last.Fm website	20 artists are displayed in the recommendation's section.

= large) are for univariate tests and the Mahalanobis distance is applied for the multivariate test. It is important to note that the strengths mentioned for the r effect are not equivalent to those of d strengths because *"Cohen requires larger effects when measuring the strength of association"* (Ellis 2009). In order to validate the directional hypotheses, we performed one-tailed tests, except for $H2$, and α was set to 0.05 for all the significance tests. The environment used for the statistical tests and preparing relevant plots was RStudio ver.0.99.892, with external packages including *Hotelling* (Curran 2015), *rrcov, mvnormtest* and *profileR* (Desjardins 2005). Effect sizes r are computed using the formulae provided by Fritz et al. (2012).

4 Execution of the Experiment

In this section, we first describe how we executed the experiment and then elaborate on the data collection steps. Figure 2 presents the flow of the events related to our

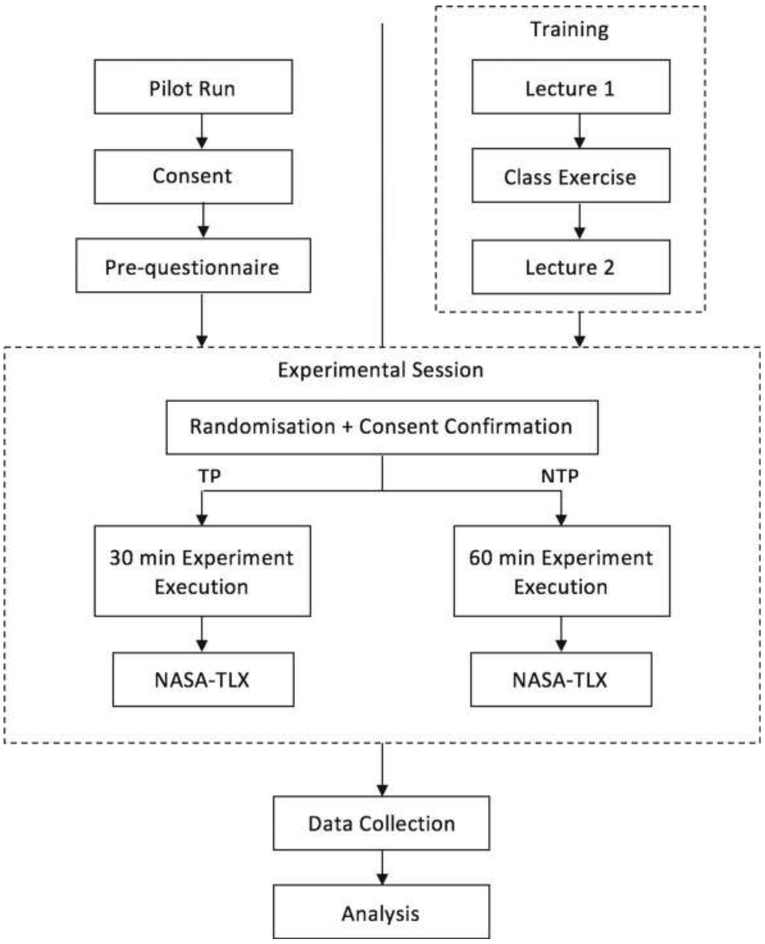


Fig. 2 Experimental process flow

experimental execution. It visualises the pre-experimental activities, experimental sessions and post-experimental activities. Figure 2 also shows which activities were executed in parallel, e.g. training was conducted in parallel to the pre-experimental activities.

4.1 Execution

The execution of the experiment involved activities taking place in two steps. In the first step, participants filled out the pre-questionnaire and signed consent forms reasonably in advance of experimental sessions taking place. However, their completion was verified before assigning the participants to the control and treatment groups on the day of the experiment.

The second step involved running the experimental sessions. The experimental sessions for TP and NTP were run in parallel in two different rooms, to which the participants were randomly assigned. On arrival, every second participant was sent to a different room to achieve a balanced design. The experimenters kept the participants unaware of the reason for their random allocation, but if a participant showed concern, s/he was informed that the activity/content would be the same in both rooms.

After randomisation, the TP session had 22 participants and the NTP session had 21 participants, totalling 43 participants. After distributing the task and the template sheets, the experimenters projected the screenshot of the UI and briefly introduced the task. The task's introductory note was the same predetermined content for both groups. We then allotted 30 minutes to the TP group and 60 min to the NTP group to complete the task and regulated the time pressure, as elaborated in Section 3.2.

We developed two detailed scripts for the TP and NTP sessions to guide the experimenters through the whole experimental session. The scripts contain the sequence of activities along with time stamps (when to say/do what), the description related to the task and instructions on how to conduct NASA-TLX. When the participants had finished with the task, post-experimental data was collected using NASA-TLX, which is a pen-and-paper activity. The actual duration of the experimental sessions did not deviate from plan.

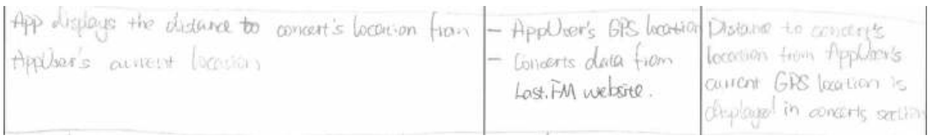
4.2 Data Collection

We marked all valid test cases designed by the participants as consistent (c) or inconsistent (ic), whereas marking the test case as *dropped* followed the criteria listed below:

- Wrong test cases: test cases where the input, expected output and the test case description were not in sync with each other, as well as test cases that resulted from the lack of understanding of the requirements. These test cases were the result of misunderstanding the specifications rather than an actual inconsistent test case.
- Specification-conflicting test cases: test cases that were in conflict with the specified requirements. In other words, a test case validating a scenario/case that cannot happen when considering the rest of the specified functionality.
- Repeated test cases: test cases that were duplicates.

Figures 3, 4, 5, 6, 7, 8 and 9 provide examples of test cases in each category. The transcribed versions, for easier readability, are available in Appendix. For MusicFone's specifications, please refer to the experimental package provided in Section 3.7.

In order to reduce subjectivity in the marking, we conducted a round of pilot marking to calculate an inter-rater agreement between the author collecting all the data and another author of the study. The pilot marking involved classifying 108 (28%) test cases of the



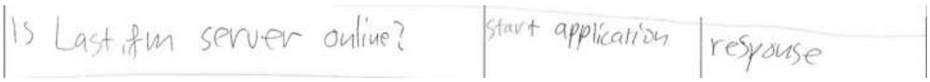
App displays the distance to concert's location from AppUser's current location	- AppUser's GPS location - Concerts data from Last.FM website.	Distance to concert's location from AppUser's current GPS location is displayed in concerts section

Fig. 3 Example of a designed consistent (c) test case

randomly chosen participants as consistent (c), inconsistent (ic) and *dropped*. We computed Randolph's free marginal kappa for inter-rater reliability, since it was not compulsory for each category to contain a certain number of cases. Randolph's free marginal kappa suggested a substantial agreement with 66% (Randolph 2005, 2008; Landis and Koch 1977). The discrepancies were resolved via discussion to establish a common understanding, and all the test cases were then marked by one of the authors. A few test cases designed by the participants could not be classified because of their confusing nature. We resolved these cases by reaching an agreement after the discussions, which further alleviated the subjectivity in marking.

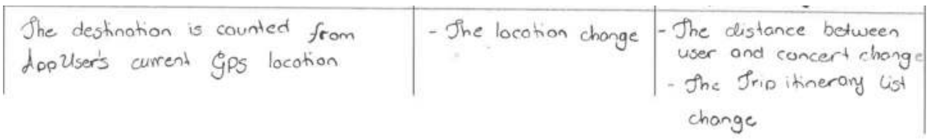
Figures 3, 4 and 5 present examples of c and ic test cases, respectively, designed by the participants. The test case in Fig. 4 is ic because the test case is validating a functionality that is not declared in the requirements. The requirements only specify the kind of data to be retrieved from the Last.Fm website and the behaviour of the MusicFone application, but the test case (Fig. 4) validates the connection between the Last.Fm server and the MusicFone application, which is certainly necessary for the application to proceed according to its specified behaviour. Figure 5 is an example of a test case that is depicting a tester's outside-of-the-box attitude, as the participant is validating the required functionality of the itinerary compilation by also considering the location changes of the application's user. Figures 6 and 7 are examples of *dropped* test cases. The test case in Fig. 6 is wrong because the data from the Last.Fm is retrieved based on the artist currently playing, not on the basis of the user's current GPS location. This shows that the participant did not develop a correct understanding of the requirements. The test case in Fig. 7 is wrong because the content in the columns are not synchronised, making the purpose of the test case ambiguous. Figure 8 shows an example of a specification-conflicting test case because it tests whether the application displays the concerts of an artist within a distance of greater than or equal to 500 km, but if that artist is added to the selected artists list, his/her concert (of >= 500 km) is not added (displayed) to the trip itinerary. It is conflicting (erroneous) because the specifications do not impose limitations based on distances, but adds it to the itinerary based on other stated conditions; hence, it was dropped.

The confusing test cases were those which were valid but difficult to classify as c or ic due to the content of the three columns in the design template. Therefore, they were resolved as either c or ic or were dropped after discussion among the authors of the study. An example of a confusing test case is shown in Fig. 9. After discussing and examining the pre-condition and expected output columns, this test case was classified as ic because



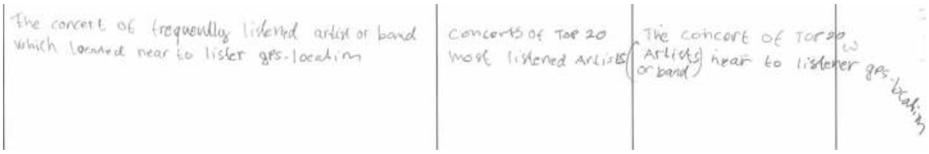
Is Last.fm server online?	Start application	response

Fig. 4 Example-1 of a designed inconsistent (ic) test case



The destination is counted from AppUser's current GPS location	- The location change	- The distance between user and concert change - The Trip itinerary list change

Fig. 5 Example-2 of a designed inconsistent (*ic*) test case

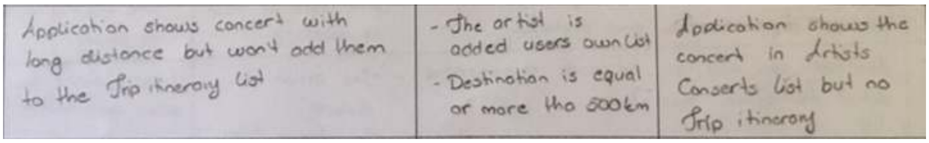


The concert of frequently listened artist or band which locatred near to lister gps-location	Concerts of Top 20 most listened Artists	The concert of Top 20 Artists (or band) near to listener gps-location	

Fig. 6 Example-1 of a designed wrong test case

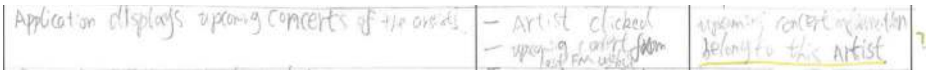
Remove the artist's from the list	- remove an artist	The artist display in the list

Fig. 7 Example-2 of a designed wrong test case



Application shows concert with long distance but won't add them to the Trip itinerary list	- The artist is added users own List - Destination is equal or more tha 500km	Application shows the concert in Artists Conserts list but no Trip itinerary

Fig. 8 Example of a designed specification-conflicting test case



Application displays upcoming concerts of the artist	- Artist clicked - upcoming concert from last FM website	upcoming concert information belong to this Artist

Fig. 9 Example of a designed confusing test case

the participant is validating that concert information fetched from the Last.Fm website truly belongs to the clicked artist.

In total, 14 out of 385 test cases were dropped from the TP and NTP groups: 7 (3.93% TP, 3.43% NTP) from each. The marking also resulted in dropping one participant from the TP group because all of his/her test cases (3 test cases) were dropped. Consequently, the number of participants in both groups became equal, i.e. 21 participants in each group, leading to a balanced design. In accordance with the measurement improvement step mentioned in Section 3.3, we added 13 inconsistent test cases from the 371 test cases designed by the participants to the predesigned set of test cases. As a result, the total number of inconsistent test cases increased to 50 IC. The total number of consistent test cases remained the same, i.e. 18 C, in the final set of test cases designed by us because no new consistent test cases were detected. Based on these data, we then calculated the rate of change values (z) of the extracted data for the TP and NTP groups.

5 Results

We first present the descriptive statistics of the collected data and then the results of hypothesis testing.

5.1 Descriptive Statistics

Table 5 shows the number of c and ic test cases, the number of participants, and the number of early finishing participants in each experimental group. In total, 174 test cases were designed by the TP group, whereas participants in the NTP group designed 197 test cases. The number of ic test cases in both groups is almost the same, but the NTP group designed 21 more c test cases than the TP group. Six out of 21 participants finished earlier than the designated time in the NTP group but none finished earlier in the TP group.

Table 6 shows the descriptive statistics for the c, ic and z values of the TP and NTP groups in terms of their minimum (min), maximum (max), mean, median (mdn) and standard deviation (sd). Descriptive statistics for the number of c and ic test cases provide insight into the design of consistent and inconsistent test cases by the participants in both experimental groups. We can see from the table that the mean of the c test cases in the TP group is less than the mean of the c test cases designed in the NTP group, i.e. slightly more consistent test cases are designed in the NTP group. The relation for the ic test cases designed in the TP and NTP groups is also similar, that is, on average, more ic test cases are designed in the NTP group than the TP group. The maximum number of c test cases

Table 5 Raw data

Group	#Participants	Type	Count	Total	#Early Finish
TP	21	c	150	174	0
		ic	24		
NTP	21	c	171	197	6
		ic	26		

Table 6 Descriptive statistics

	Group	mean	mdn	min	max	sd
c	TP	7.143	7.000	3.000	11.000	2.032
	NTP	8.143	8.000	3.000	15.000	2.670
ic	TP	1.143	0.000	0.000	8.000	2.056
	NTP	1.238	1.000	0.000	6.000	1.670
z	TP	0.374	0.389	0.062	0.611	0.133
	NTP	0.428	0.424	0.107	0.833	0.166

in the NTP group (15) is greater than the maximum number of c test cases in the TP (11) group, which is expected since the NTP group had more time to work on the task. The standard deviation of c and ic are similar in both groups. Considering the z values, the mean of the NTP group is greater than the mean of the TP group with similar variation, which surprisingly suggests relatively more confirmatory behaviour in the absence of time pressure.

The box plots in Fig. 10 present the spread of the c and ic data of the TP and NTP groups. We can see that the median of c in the NTP group is greater than the median of c in the TP group and similar to the case of the minimum and maximum values of c in both groups. The median of ic in the NTP and TP groups follows the same trend, however, the minimum value is 0 in both groups. The maximum value of ic in the TP group is greater than the maximum value of ic in the NTP group.

Due to the scale difference between the measurements of absolute counts and the z values, we present the box plots of z in the TP and NTP groups in Fig. 11. We can see in Fig. 11 that there are no outliers in the TP group, whereas there are three in the NTP group,

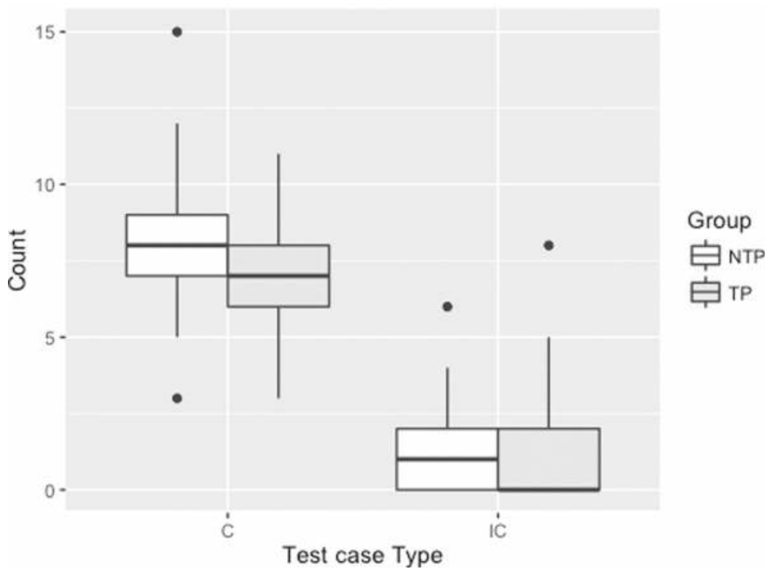


Fig. 10 c and ic data of the TP and NTP groups

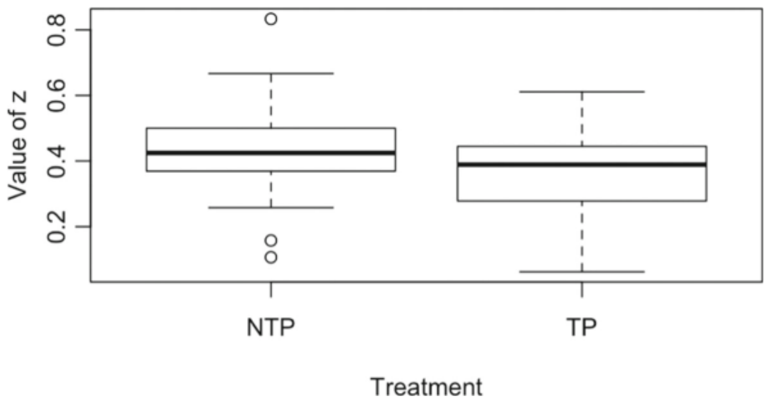


Fig. 11 Rate of change (z)

affecting the descriptive statistics, e.g. the mean value. However, there is a slight difference between the median values of the two groups. The minimum value of the NTP group is clearly greater than the minimum value of the TP group. Finally, the spread of the values in the first quartile and the fourth quartile of both groups are almost equal to each other.

The z values for both groups reveal that all the participants designed relatively more consistent test cases because all the values of z are greater than 0. Moreover, none of the participants have designed a *relatively* equal number of c and ic test cases in either group, as minimum the value of z is 0.062 (TP). However, when we examine the absolute counts, there are a few participants who designed an almost equal number of c (consistent) and ic (inconsistent) test cases. In general, the number of inconsistent test cases in both groups are remarkably few when compared to the consistent cases. Additionally, the absolute counts of c and ic suggest that there are more participants in the TP group who have not designed any inconsistent test cases at all. The descriptive statistics show that on average, participants in the NTP group designed comparatively more test cases. This indicates that the participants in the NTP group gave more coverage to the test cases as compared to the TP group - which draws attention to the fact that the NTP group had relatively more time than the TP group. However, it is interesting that the coverage in the NTP group is not exceptionally more than in the TP group. It is also evident from the descriptive statistics that despite the time pressure, participants in both groups designed more consistent test cases than inconsistent test cases.

Figure 12 presents the box plots of the temporal demand attribute of NASA-TLX (time pressure perceived by the participants) for the TP and NTP groups. There is a major difference between the medians of the TP and NTP groups, i.e. 83 points and 55 points, respectively. However, the minimum value for both groups is the same, i.e. 15 points, but the maximum value of the TP group (100 points) is much higher than the maximum value of the NTP group (75 points) which is even less than the median of the TP group (83 points). These box plots suggest that the temporal demand perceived by the TP group is much greater than the temporal demand perceived by the NTP group when performing the same task.

5.2 Hypothesis Testing

In this section we present the results of hypothesis testing for $H1$, $H2$, $H3$ and $H4$.

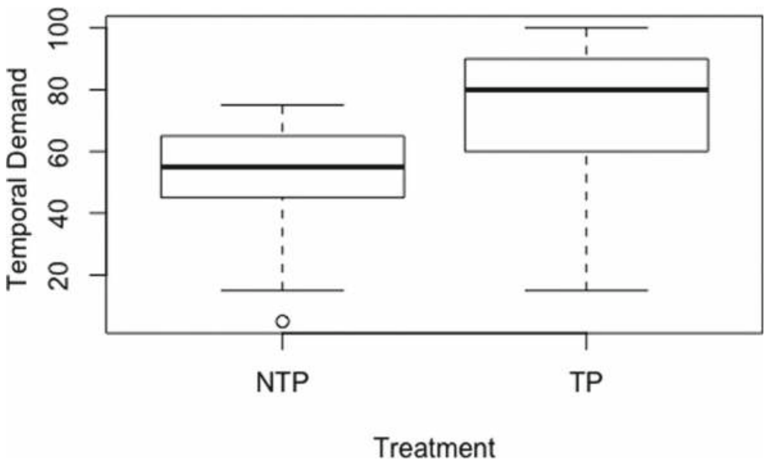


Fig. 12 NASA-TLX temporal demand

5.2.1 Hypothesis 1

In order to test $H1$, we first test it for the pooled data of the TP and NTP groups. For further analysis, we perform statistical testing on the TP and NTP data separately, as well. The normality assumption for the pooled data was tested for c and ic, where ic data failed to satisfy with $p{-}value = 5.987e{-}08$. Similarly, the TP and NTP groups also failed to satisfy the assumption of normality of their ic data, with $p - values$ of $4.727e - 06$ and $2.06e - 4$, respectively. Therefore, significance testing was performed with the Wilcoxon signed-rank test (a non-parametric variant of the t-test) for all these cases. We applied the Bonferroni adjustment $\alpha = 0.016$ to the pooled case and for TP and NTP testing. The null hypothesis is rejected with a $p - value = 2.161e - 08$, an effect size $r = 0.598$ (large) and a $df = 41$ for the pooled data. The null hypothesis for NTP was rejected with a $p - value = 5.48e - 05$ and the effect size r is 0.600, which is large. The test on the TP data also rejected the null hypothesis with a $p - value = 5.8e - 05$ and an effect size $r = 0.597$, which is again large, and df is the same for both statistical tests for the NTP and TP, i.e. 20.

As a result, we reject $H1_0$ and find evidence to support that testers design more consistent test cases than inconsistent test cases.

5.2.2 Hypothesis 2

We performed a multivariate test of the mean differences - Hotelling's T^2 for the TP and NTP groups for the two dependent variables, c and ic, after checking the test's assumptions. Note that z was not included in this test because of its high correlation with c and negative correlation with ic, which could violate the multivariate normality assumption of this test. Hence, we statistically tested z separately. The first two assumptions of Hotelling's T^2 hold. The results of Bartlett's test revealed that both dependent variabless, c and ic, satisfy the assumption of a common variance-covariance matrix with $p - values = 2.3e - 1$ $(df = 1)$ and $3.6e - 1$ $(df = 1)$, respectively. Additionally, the last assumption of multivariate normality was satisfied after performing natural log transformation on the ic data of the TP and NTP groups with $p - values = 3.5e - 1$ and $3.26e - 1$, respectively.

Hotelling's T^2 test was unable to detect a statistically significant difference between the two groups, i.e. we fail to reject $H2_0$, $T^2 = 5.012$, $F(2, 39) = 2.44$, $p - value = 1.001e - 1$, $Mahalanobis\ D = 0.691$ and variance is $\eta = 0.111$ with a null confidence interval. The result of this test suggests that time pressure may not have an effect on the (dis)confirmatory behaviour of testers.

Figure 13 presents the profiles of c and ic, which implies that they may not be parallel. Furthermore, null hypotheses testing of whether the profiles are parallel is rejected with $p - value = 4.145709e - 01$, suggesting an interaction between the variables. This might imply that in the NTP group, participants designed more consistent test cases; however, it can be due to the floor effect of the low number of ic in both groups. $H3$ proceeds with an additional analysis of this.

5.2.3 Hypothesis 3

To determine the effect of time pressure on confirmation bias in relative terms, we first performed the normality tests on the z data of both groups, and the results revealed that our data are normally distributed with a $p - value = 2.54e - 1$ for the TP group and a $p - value = 3.52e - 1$ for the NTP group. Therefore, we executed two-sample t-test that failed to reject the null hypothesis with a $p - value = 8.72e - 1$ and a 95% confidence interval is $(-0.131, Inf)$. Degree of freedom is 40 and the effect size d is -0.357, indicating that the effect is small, although the confidence interval contains 0 and it decreases the mean in the direction of the TP group.

5.2.4 Hypothesis 4

To test the sanity check hypothesis, which was developed to validate the operationalisation of time pressure, we first inspected the normality of the data for the TP and NTP groups.

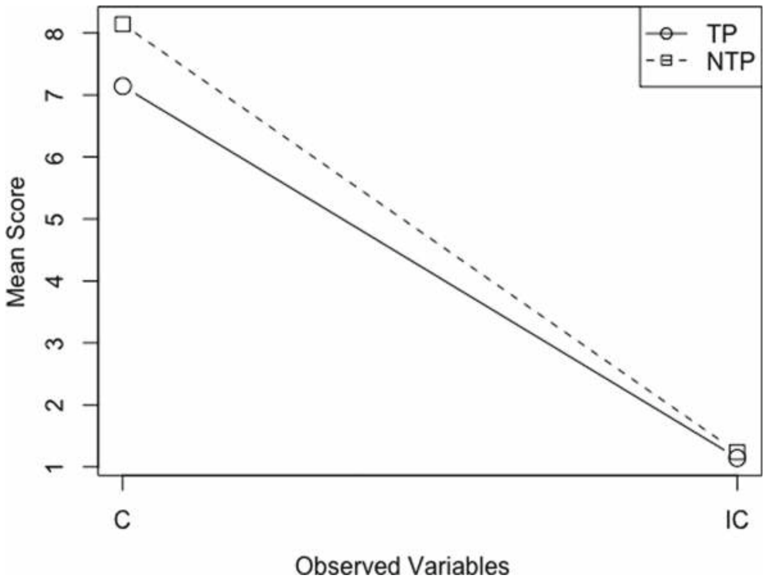


Fig. 13 Profile Plot

For this, we considered the data of the temporal demand attribute from the NASA-TLX scaled measurements. The normality tests revealed that the data for the TP and NTP groups do not follow a normal distribution. Hence, we performed a two-sample Mann-Whitney test yielding a $p - value = 1.231e - 3$, thus we reject the null hypothesis, revealing that time pressure or temporal demand perceived by the TP group is significantly more than the temporal demand perceived by the NTP group. The effect size r is −0.469, which is medium, and $df = 40$.

5.3 Summary of Results

Table 7 presents a summary of the hypothesis testing results. The effect size column lists the effect size along with its type, e.g. r. We can see that $H1_0$ is rejected, signifying that consistent test cases are designed more often than inconsistent test cases. We fail to reject $H2_0$, which shows that time pressure may not significantly affect the (dis)confirmatory behaviour. The case is similar with the results of $H3$: we did not observe the manifestation of confirmation bias at different rates between the TP and NTP groups. These results need further investigation with a larger sample size. However, we are confident in the successful operationalisation of time pressure, which is evident from the results of $H4_0$.

$H1$ results provide evidence that testers exhibit a confirmatory behaviour while designing test cases, i.e. they designed considerably more consistent test cases than inconsistent ones. This suggests that *participants were highly biased in their approach towards designing test cases*. On the other hand, comparison of the experimental groups for observing the effect of time pressure in terms of the designed test cases, $H2$, and with the rate of change aspect, $H3$, did not reveal any evidence for an effect due to time pressure. Specifically, *in our experiment, time pressure did not cause the participants to manifest more confirmation bias either in absolute terms or relative terms*. On the other hand, confirmation bias has *always* manifested, regardless of time pressure, in the designing of test cases. The hypothesis that we developed to validate the values of our levels of time pressure ($H4$) revealed that the time pressure perceived by the TP group was significantly more than the time pressure perceived by the NTP group. Conclusively, the results of this hypothesis suggest that we successfully manipulated time pressure for the TP and NTP groups. Despite the use of

Table 7 Hypothesis results summary

Hypothesis	p-value	Effect size	df	Decision
$H1_0$	$2.161e - 08$	0.598 $(r - large)$	41	rejected
	$5.48e - 05$	0.600 $(r - large)$	20	
	$5.8e05$	0.597 $(r - large)$	20	
$H2_0$	$1.001e - 1$	0.691 $(Mahalanobis D)$	(2, 39)	failed to reject
$H3_0$	$8.72e - 1$	−0.357 $(d - small)$	40	failed to reject
$H4_0$	$1.231e - 3$	−0.469 $(r - medium)$	40	rejected

NASA-TLX for measuring the task difficulty as perceived by the task performer, to our knowledge there is no standard way to interpret the findings.

6 Discussion

The reasons for these results are manifold. The first possible reason is the object of our study, MusicFone, which is a ***realistic and complex object***. Accordingly, the participants may have faced problems in designing test cases for the object despite being provided with the conceptual prototype (screenshot) to help understand it. This is partly supported with the observation that the coverage of specifications is low not only for inconsistent test cases but also for consistent ones. Additionally, we observed that no large differences existed in the coverage between the TP and NTP groups. It is conceivable that the usage of a toy object in the experiment might have influenced the results differently, though such an approach would have compromised the realism aspect.

Besides the complexity of the task, one other reason for the results could be the participants' ***unfamiliarity with the domain*** of the provided object. If the participants had been trained in or were experienced in a similar domain to that of MusicFone, it might have also produced relatively diverse results. But in such a case, imposing similar time limits to create time pressure would be unjustifiable, as their familiarity with the domain might not have allowed them to experience the time pressure to its full extent. Therefore, to observe the effect of domain, care has to be taken in deciding the duration when creating time pressure.

Another reason for the results can be participants' ***(in)experience in testing***, as investigated by Calikli and Bener (2010, 2014) and Teasley et al. (1994), because there was room for the participants to design more inconsistent test cases, especially the NTP group. Yet, our participants can be considered novice professionals because most had less than three years of industrial experience. The chances that our results are affected by the given ***training on equivalence partitioning strategy*** are low. In equivalence partitioning, test cases are designed based on the identification of valid and invalid classes. We considered a test case consistent if the functionality was explicit in the specifications and inconsistent otherwise, irrespective of the class from which the test case was derived. The same concept of (dis)confirmatory behaviour applies to the software testers in industry as well. Regardless of the strategy they follow in designing test cases (e.g. equivalence partitioning, boundary testing), if the specifications are explicit about a functionality then a test case validating the respective specification is consistent (confirmatory) and inconsistent (disconfirmatory) otherwise.

Unfamiliarity with the domain, together with limited experience, may have further enhanced the effect of task complexity. This may have made the time alloted inadequate for the control group as well. As a result, no significant differences existed in the (dis)confirmatory behaviour between the groups due to time pressure. The participants challenged with completing the task within the available time might have kept on exploring and understanding the complex object in parallel to designing test cases. This may have caused the designing of more consistent test cases as compared to inconsistent ones in both groups. The total number of test cases designed by each of the 6 participants who finished early in the NTP group range from a minimum of 6 to a maximum of 15 test cases in the NTP group. The maximum number achieved by an early finisher indicates that s/he designed all the possible test cases. In other words, s/he could not think of any more test cases and thus finished before the end of the allotted time for the NTP group. The minimum number of test cases (in the NTP group) achieved by the group of early finishers possibly indicates an inclination to quit. Both situations depict less experienced participants handling a complex task. However, the data of the six participants who finished early is not enough to support the presumption of added task complexity. One reason for the relatively low number of inconsistent test cases in both groups could be ***no feeling of accomplishment***. Pham et al. (2014) found this aspect to be an inhibiting factor with respect to testing. The feeling of accomplishment comes from the detection of actual defects, but since students are novice testers, they do not have this experience (Pham et al. 2014). Pham et al. (2014) investigated the enablers, inhibitors and perceptions of testing by examining student projects that involved development. In relation to our study, the participants might have also had an inadequate feeling of accomplishment because of their lack of considerable or industrial experience with testing. This inadequate feeling could also be due to the fact that our participants did not execute the test cases, since they were only limited to designing them. If they could have executed the test cases, especially the inconsistent ones, defect detection (if any) could have motivated them to design and execute more inconsistent test cases, thereby enriching them with the feeling of accomplishment. However, their interaction with the application (MusicFone) for the execution of the test cases and the presence of defects could have created construct and internal validity threats. As mentioned in Section 3.7.2, the presence of or feedback from errors can possibly affect the testing behaviour in terms of a positive test bias (Teasley et al. 1994).

The ***completeness of specifications*** is also an important dimension with respect to the object used in our study. Complete requirements suggest that every required and non-required behaviour is stated in explicit terms. In such a scenario, test cases designed to confirm the requirements are all consistent test cases, which is a depiction of confirmatory behaviour or a manifestation of confirmation bias. Manifestation of confirmation bias with complete specifications is then not adverse in software testing because every required and non-required behaviour is confirmed via testing. Our object, having a realistic level of completeness, provided the participants with an opportunity to design inconsistent test cases, including outside-of-the-box cases. Nevertheless, participants possibly struggled due to their inexperience, domain unfamiliarity and lack of feeling of accomplishment, which limited them in designing substantial inconsistent test cases. Our results support the findings of Causevic et al. (2013), Leventhal et al. (1994) and Teasley et al. (1994) that the participants exhibited a significant confirmatory behaviour in designing the test cases. The results are also similar to the findings of Mäntylä et al. (2014), Nan and Harter (2009) and Topi et al. (2005), as time pressure did not affect the dependent variable. The results of our study also challenge the results of Leventhal et al. (1993) and Leventhal et al. (1994), as they interpreted the results with respect to valid (*positive*) and invalid (*negative*) equivalence classes. The results of these studies might differ if the test cases are reassessed according to (in)consistent terminology. In the case of Causevic et al. (2013), it is unclear how they distinguished between the test cases as *positive* and *negative*, despite defining the terms; *"the experiment performed in this study has a built-in mechanism of differentiating whether a particular test case is of a positive or negative type*" (Causevic et al. 2013, p. 3).

The testers' confirmatory behaviour in designing test cases may compromise the quality of testing which consequently may deteriorate software quality. It is harmful because testers tend to confirm the functionality of the program rather than test it to reveal its weaknesses and defects. In other words, they do not test all aspects of the functionality, e.g. corner cases. This confirmatory approach may contribute to a low external quality, thus elevating development and testing costs. As mentioned earlier, the phenomenon of confirmation bias with complete specifications may not have a deleterious impact. Yet, it may become problematic due to the prevalence of incomplete requirements specifications in the SE industry that fail to elicit every functionality, especially for large and complex systems

(Albayrak et al. 2009; Paetsch et al. 2003). Software testers design test cases based on their understanding of (often incomplete) requirements specifications, which may lead to the incomplete coverage of exceptional cases: inconsistent test cases. Therefore, it is essential that the testers develop a disconfirmatory *outside-of-the-box* attitude towards software testing. Disconfirmatory attitude development should also be taught to software engineering students alongside the teaching of standard testing techniques, e.g. equivalence partitioning. Furthermore the degree to which time pressure promotes the confirmatory attitude among testers remains inconclusive, as we have not detected statistically significant evidence to show whether this is present.

7 Threats to Validity

In this section, we analyse the threats to the validity of our study and discuss the relevant ones, following the list provided by Wohlin et al. (2000).

7.1 Construct Validity

The interaction of task and treatment, due to a one-factor two-treatment experimental design with one object, is a shortcoming of our study in terms of the mono-operation bias threat. This caused the results to be limited to the single object (MusicFone) alone. Another possible threat is related to the total count of inconsistent (IC) test cases. In the validity improvement step, we increased the IC count through the addition of 13 test cases, and it is possible that this count would further increase if this study is replicated. One possible effect of this could be the observation of results that are more in favour of confirmatory behaviour or confirmation bias manifestation, since the higher the IC value, the lower the coverage of inconsistent test cases. The mono-method bias threat was addressed by executing a pilot run and through the resolution of the confusing test cases. Our study is not prone to the threat of testing and treatment interaction because the participants were asked to perform the experimental task as if it were a normal class activity; additionally, they were unaware of the treatment that could impact the results. The example test case provided in the test case design template was a low-hanging fruit. Although it was a consistent test case example, we do not think that it biased the participants to design either consistent or inconsistent test cases.

7.2 Internal Validity

Here, we discuss the threats to internal validity that are related to multiple groups because our study involved two groups. In our study, we taught and trained all the participants together and randomly assigned them to the experimental and control groups. In this way, we avoided the interaction with selection threat. In our opinion, the provision of a conceptual prototype in the experimental run, that was different from the training, did not influence the results because it was provided to both the experimental and control groups. Moreover, we applied the treatment to the experimental group in parallel to the control group, but in a different room; thus, we prevented the imitation of treatments threat. We addressed the threat of compensatory equalisation of treatment by not compensating either the treatment or controlled group in a special way. Additionally, we improved the instrumentation as a result of the pilot run and we also enhanced the baseline test case set from the participants' test case data. Moreover, since all the participants underwent the same training, this precluded compensatory rivalry or resentful demoralisation. Regarding the experimental designs, we did not choose other designs because of the limitations of the course's time frame. The announcement of bonus marks as an incentive to participate in the experiment was not a threat to the internal validity because the incentive was neither coercive nor constituted undue influence (Commission 2012).

7.3 External Validity

We can categorise our participants as a proxy for novice professionals based on their experience, as discussed in Section 3.6. However, the use of student participants rather than professionals poses a selection and treatment interaction threat to the external validity, for they were first-year graduate students, with many having less than six months of industrial experience. Moreover, we conducted the study in an academic environment and manipulated time pressure in a controlled and psychological manner, i.e. the other factors that could affect the application of the treatment were not prominently present. However, we used a realistic object for our study; therefore, we tried to achieve realism for the task. Additionally, we are not of the opinion that the use of pen and paper for performing the task is unrealistic because our study focused on the designing phase of test cases rather than their execution. Conclusively, we have addressed the interaction of the setting and treatment threat to the best possible extent considering the context of the experiment.

7.4 Conclusion Validity

It is possible that we are committing a Type II error by failing to reject the null hypotheses ($H2_0$, $H3_0$) related to time pressure's effect on confirmation bias manifestation, because the effect size indicates there might be an effect, albeit small. This needs to be addressed via replications with larger samples. We addressed the threat of violated assumptions by testing our data for them before validating our hypothesis statistically. We have addressed the reliability of measures threat by executing a pilot run and computing an inter-rater agreement for evaluating test cases as (in)consistent. This established a common understanding among the authors of the study and avoided taking the subjective view of a single evaluator alone. The reliability of treatment threat was addressed by preparing a scripted guideline for the experimental and control groups to help the experimenters implement the treatment and control uniformly and through the sanity check hypothesis.

8 Conclusion

In this study, we examined the manifestation of confirmation bias in a functional software testing context and considered whether - if such a manifestation exists - it is affected by time pressure. In order to address our goals, we executed an experiment with 42 graduate (master's degree) students by manipulating the time pressure in the treatment and control groups. We have detected the manifestation of confirmation bias in terms of the specification-consistent and specification-inconsistent test cases. Therefore, our study contributes to the existing body of literature by providing empirical evidence in support of the confirmatory behaviour in functional software testing and the investigation of time pressure's effect on confirmation bias in SE.

Pertaining to the observed results, the answer to *RQ1* is that our study documents the confirmation bias manifestation, in other words, confirmatory behaviour, in designing functional test cases, similar to previous research findings (Causevic et al. 2013; Leventhal et al. 1994; Teasley et al. 1994). Test designers generated significantly more, and with a large effect size, specification-consistent test cases. For *RQ2*, however, we do not have evidence for an effect on confirmatory behaviour due to time pressure.

In terms of our recommendations to practitioners, we should revisit the terms *consistent* and *inconsistent* test cases in the context of functional software testing. Our definition of a consistent test case in relation to confirmation bias suggests that confirmation bias may not always be a negative phenomenon, provided that the requirements specifications are *complete* and well-elicited in terms of the required and non-required functionality or behaviour. However, this is not usually the case in the fast-paced software industry. Therefore, the reality is that in practice, confirmation bias is more likely a phenomenon that adversely affects software quality through limiting the verification of unspecified, yet essential, software behaviour. While it would be unrealistic to recommend changes to specification elicitation practices, we can make recommendations targeted at individuals working in the testing profession. We recommend that testers develop self-awareness of confirmation bias and counter it with a disconfirmatory attitude.

For researchers, the results of our study suggest that there is a need to replicate or extend this experiment with larger samples in order to detect the effect of time pressure, if any, and to investigate other possible factors that might have an impact on confirmation bias or can possibly mitigate the effect of time pressure on confirmation bias. For example, task complexity, domain familiarity and experience as confounding factors are among the possible extensions for future work. A qualitative study is also planned that would help in identifying whether software testers prefer confirmatory test cases to begin testing and what factors (e.g. time pressure) influence their preferences and behaviours.

Acknowledgements Open access funding provided by University of Oulu including Oulu University Hospital. The authors would like to thank the associate editor and three anonymous reviewers for their suggestions on earlier versions of this paper. This study was supported in part by the Infotech Oulu Doctoral grant at the University of Oulu to Iflaah Salman.

Appendix: Transcribed Test Cases

Figure	Description	Input / Pre-condition	Expected output / Post-condition
Fig. 3	App displays the distance to concert's location from AppUser's current location	- AppUser's GPS location - Concerts data from Last.FM website.	Distance to concert's location from AppUser's current GPS location is displayed in concerts section.
Fig. 4	Is Last.fm server online?	start application	response

Figure	Description	Input / Pre-condition	Expected output / Post-condition
Fig. 5	The destination is counted from AppUser's current GPS location	- The location change	-The distance between user and concert change -The trip itinerary list change
Fig. 6	The concert of frequently listened artist or band which located near to listener gps-location	concerts of top 20 most listened Artists (Artists or band)	The concert of top 20 near to listener gps-location
Fig. 7	Remove the artists from the list	- remove an artist	The artist display in the list
Fig. 8	Application shows concert with long distance but won't add them to the trip itinerary list	- The artist is added users own list - Destination is equal or more than 500 km	Application shows the concert in artists Concert list but no trip itinerary
Fig. 9	Application displays upcoming concerts of the artists	- Artists clicked - upcoming concert from Last.FM website.	upcoming concert information belong to this artist.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

Albayrak, Kurtoglu H, Bçakç M (2009) Incomplete software requirements and assumptions made by software engineers. In: Proceedings of 16th asia-pacific software engineering conference, pp 333–339, https://doi.org/10.1109/APSEC.2009.39

Arnott D (2006) Cognitive biases and decision support systems development: a design science approach. Inf Syst J 16(1):55–78. https://doi.org/10.1111/j.1365-2575.2006.00208.x

Ask K, Granhag PA (2007) Motivational bias in criminal investigators' judgments of witness reliability. J Appl Soc Psychol 37(3):561–591. https://doi.org/10.1111/j.1559-1816.2007.00175.x

Austin RD (2001) The effects of time pressure on quality in software development : an agency model. Inf Syst Res 12(2):195–207

Baddoo N, Hall T (2003) De-motivators for software process improvement: an analysis of practitioners' views. J Syst Softw 66(1):23–33

Basili VR, Caldiera G, Rombach DH (1994) The Goal Question Metric Approach. Encyclopedia of Software Engineering 1, https://www.bibsonomy.org/bibtex/2ed38a7a0ec5148979dc72d0a65034eb4/stammel

Calikli G, Bener A (2010) Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance. In: Proceedings of the 6th international conference on predictive models in software engineering - PROMISE '10, https://doi.org/10.1145/1868328.1868344, http://portal.acm.org/citation.cfm?doid=1868328.1868344

Calikli G, Bener A (2014) Empirical analysis of factors affecting confirmation bias levels of software engineers. Softw Qual J, https://doi.org/10.1007/s11219-014-9250-6, http://link.springer.com/10.1007/s11219-014-9250-6

Calikli G, Bener AB (2013) Influence of confirmation biases of developers on software quality: an empirical study. Softw Qual J 21(2):377–416. https://doi.org/10.1007/s11219-012-9180-0. http://link.springer.com/10.1007/s11219-012-9180-0

Calikli G, Arslan B, Bener A (2010a) Confirmation bias in software development and testing : an analysis of the effects of company size, experience and reasoning skills. In: 22nd annual psychology of programming interest group workshop

Calikli G, Bener A, Arslan B (2010b) An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers. In: 2010 ACM/IEEE 32nd international conference on software engineering, vol 2, pp 187–190, https://doi.org/10.1145/1810295.1810326

Calikli G, Bener A, Aytac T, Bozcan O (2013) Towards a metric suite proposal to quantify confirmation biases of developers. International Symposium on Empirical Software Engineering and Measurement: 363–372. https://doi.org/10.1109/ESEM.2013.47

Cataldo M (2010) Sources of errors in distributed development projects: implications for collaborative tools. In: Proceedings of the 2010 ACM conference on computer supported cooperative work p 281–290, https://doi.org/10.1145/1718918.1718971. http://portal.acm.org/citation.cfm?id=1718971

Causevic A, Shukla R, Punnekkat S, Sundmark D (2013) Effects of negative testing on tdd: an industrial experiment. In: International conference on agile software development, pp 91–105. http://link.springer.com/chapter/10.1007/978-3-642-38314-4_7

Commission NBA (2012) Ethical and policy issues in international research: clinical trials in developing countries. https://doi.org/10.1109/UPEC.2015.7339896, https://bioethicsarchive.georgetown.edu/nbac/clinical/Vol1.pdf

Curran JM (2015) Package Hotelling. Tech. rep.., https://cran.r-project.org/web/packages/Hotelling/Hotelling.pdf

Davis A, Overmyer S, Jordan K, Caruso J, Dandashi F, Dinh A, Kincaid G, Ledeboer G, Reynolds P, Sitaram P, Ta A, Theofanos M (1993) Identifying and measuring quality in a software requirements specification. In: First international software metrics symposium

Desjardins CD (2005) Journal of statistical software profile analysis of multivariate data in r : an introduction to the profileR package. J Stat Softw VV(II):1–29

Ellis P (2009) Thresholds for interpreting effect sizes. http://www.polyu.edu.hk/mm/effectsizefaqs/thresholds_for_interpreting_effect_sizes2.html

Fritz CO, Morris PE, Richler JJ (2012) Effect size estimates: current use, calculations, and interpretation. J Exp Psychol Gen 141(1):2–18. https://doi.org/10.1037/a0024338. http://www.ncbi.nlm.nih.gov/pubmed/21823805

Fucci D, Turhan B, Juristo N, Dieste O, Tosun-Misirli A, Oivo M (2015) Towards an operationalization of test-driven development skills: an industrial empirical study. Inf Softw Technol 68:82–97

Gilovich T, Griffin D, Kahneman D (2002) Heuristics and biases, 8th edn. Cambridge University Press, Cambridge

Harris ML, Collins RW, Hevner AR (2009) Agile methods: fast-paced, but how fast? In: 15th Americas conference on information systems 2009, AMCIS 2009 vol 2, pp 1020–1026, http://www.scopus.com/inward/record.url?eid=2-s2.0-84870160482&partnerID=tZOtx3y1

Hazzan O, Hazzan O, Dubinsky Y, Dubinsky Y (2007) The software engineering timeline : a time management perspective. In: International conference on software-science, technology & engineering, 2007. SwSTE 2007. IEEE, pp 95–103. https://doi.org/10.1109/.9

Hernandez I, Preston JL (2013) Disfluency disrupts the confirmation bias. J Exp Soc Psychol 49(1):178–182. https://doi.org/10.1016/j.jesp.2012.08.010. http://linkinghub.elsevier.com/retrieve/pii/S002210311200176X

Hulland JS, Kleinmuntz DN (1994) Factors influencing the use of internal summary evaluations versus external information in choice. J Behav Decis Mak 7(2):79–102

Klayman J, Ha YW (1989) Hypothesis testing in rule discovery: strategy, structure, and content. J Exp Psychol Learn Mem Cogn 15(4):596–604. https://doi.org/10.1037/0278-7393.15.4.596

Kuutila M, Mantyla MV, Claes M, Elovainio M (2017) Reviewing literature on time pressure in software engineering and related professions: computer assisted interdisciplinary literature review. In: Proceedings - 2017 IEEE/ACM 2nd international workshop on emotion awareness in software engineering, SEmotion 2017, pp 54–59. https://doi.org/10.1109/SEmotion.2017.11

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data data for categorical of observer agreement the measurement. Biometrics 33(1):159–174. https://www.dentalage.co.uk/wp-content/uploads/2014/09/landis_jr__koch_gg_1977_kappa_and_observer_agreement.pdf

Leventhal LM, Teasley B, Rohlman DS, Instone K (1993) Positive test bias in software testing among professionals: a review. In: Human-Computer Interaction, pp 210–218. https://doi.org/10.1007/3-540-57433-6_50

Leventhal LM, Teasley BE, Rohlman DS (1994) Analyses of factors related to positive test bias in software testing. Int J Hum Comput Stud 41:717–749

Mäntylä MV, Itkonen J (2013) More testers-The effect of crowd size and time restriction in software testing. Inf Softw Technol 55(6):986–1003

Mäntylä MV, Petersen K, Lehtinen TOa, Lassenius C (2014) Time pressure: a controlled experiment of test case development and requirements review. In: Proceedings of the 36th international conference on software engineering - ICSE 2014, pp 83–94. https://doi.org/10.1145/2568225.2568245. http://dl.acm.org/citation.cfm?doid=2568225.2568245

Mohanani R, Salman I, Turhan B, Rodriguez P, Ralph P (2018) Cognitive biases in software engineering : a systematic mapping study. IEEE Trans Softw Eng: 1–25. https://doi.org/10.1109/TSE.2018.2877759

Nan N, Harter DE (2009) Impact of budget and schedule pressure on software development cycle time and effort. IEEE Trans Softw Eng 35(5):624–637

Nickerson RS (1998) Confirmation bias: a ubiquitous phenomenon in many guises. Rev Gen Psychol 2(2):175–220. https://doi.org/10.1037/1089-2680.2.2.175

Paetsch F, Eberlein A, Maurer F (2003) Requirements engineering and agile software development. In: Proceedings of the 12th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises (WETICE'03), pp 1–6

Pham R, Kiesling S, Liskin O, Singer L, Schneider K (2014) Enablers, inhibitors, and perceptions of testing in novice software teams categories and subject descriptors. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014, pp 30–40

Randolph JJ (2005) Free-marginal multirater kappa. In: Joensuu University learning and instruction symposium 2005

Randolph JJ (2008) Online Kappa Calculator. http://justusrandolph.net/kappa/#dInfo

Research Group HP, Ames Research Center N (1987) NASA Task Load Index (TLX)

Salman I (2016) Cognitive biases in software quality and testing. In: Proceedings - international conference on software engineering

Salman I, Misirli AT, Juristo N (2015) Are students representatives of professionals in software engineering experiments? Proceedings - international conference on software engineering 1:666–676. https://doi.org/10.1109/ICSE.2015.82

Shah H, Harrold MJ, Sinha S (2014) Global software testing under deadline pressure: vendor-side experiences. Inf Softw Technol 56(1):6–19

Sjoberg D, Anda B, Arisholm E, Dyba T, Jorgensen M, Karahasanovic A, Koren E, Vokac M (2002) Conducting realistic experiments in software engineering. In: Proceedings international symposium on empirical software engineering, pp 17–26. https://doi.org/10.1109/ISESE.2002.1166921

Stacy W, MacMillan J (1995) Cognitive bias in software engineering. Commincations of the ACM 38(6):57–63. https://doi.org/10.1007/978-3-319-08581-4

Teasley BE, Leventhal LM, Mynatt CR, Rohlman DS (1994) Why software testing is sometimes ineffective: two applied studies of positive test strategy. J Appl Psychol 79(I):142. https://doi.org/10.1037/0021-9010.79.1.142

Topi H, Valacich JS, Hoffer JA (2005) The effects of task complexity and time availability limitations on human performance in database query tasks. Int J Hum Comput Stud 62(3):349–379. https://doi.org/10.1016/j.ijhcs.2004.10.003

Tversky A, Kahneman D (1973) Judgement under uncertainty: heuristics and biases. Tech. rep., Oregon Research Institute Research Bulletin, https://doi.org/10.1126/science.185.4157.1124

Wason PC (1960) On the failure to eliminate hypotheses in a conceptual task. Q J Exp Psychol 12(3):129–140. https://doi.org/10.1080/17470216008416717. http://www.tandfonline.com/doi/abs/10.1080/17470216008416717

Wilson DN, Hall T (1998) Perceptions of software quality: a pilot study. Softw Qual J 7(1):67–75. http://link.springer.com/article/10.1023/B:SQJO.0000042060.88173.fe

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction, vol 15. Springer, Netherlands. https://doi.org/10.1007/978-3-642-29044-2

Iflaah Salman received her MSc in Information Processing Science (2014) from the University of Oulu, Oulu, Finland. She continued as a PhD candidate at the same institution with M3S Research Unit on Infotech Oulu Doctoral Research Grant. Her research interests include empirical software engineering, cognitive aspects and software testing. Previously, she worked as a software quality assurance engineer (2010 to 2012) at i2c inc., Lahore, Pakistan. She is a member of ACM and IEEE.

Burak Turhan PhD (Bogazici), is an Associate Professor in Cyber Security & Systems at Monash University. His research focuses on empirical software engineering, software analytics, quality assurance and testing, human factors, and (agile) development processes. Dr. Turhan has published over 100 articles in international journals and conferences, received several best paper awards, and secured research funding exceeding 2 million euros. He has served on the program committees of over 30 academic conferences, on the editorial or review boards of several journals including *IEEE Transactions on Software Engineering, Empirical Software Engineering* and *Journal of Systems and Software*, and as (co-)chair for PROMISE'13, ESEM'17, and PROFES'17. He is currently a steering committee member for PROMISE and ESEM, and a member of ACM, ACM SIGSOFT, IEEE and IEEE Computer Society. For more information please visit: https://turhanb.net.

Sira Vegas received her PhD degree from the Universidad Politécnica de Madrid in 2002. She is currently associate professor of software engineering at Universidad Politécnica de Madrid. Her main research interests are experimental software engineering and software testing. She is regular reviewer of highly ranked journals such as IEEE Transactions on Software Engineering, Empirical Software Engineering Journal, ACM Transactions on Software Engineering and Methodology and Information and Software Technology. Dr. Vegas was program chair for the International Symposium on Empirical Software Engineering and Measurement (ESEM) in 2007. She began her career as a summer student at the European Centre for Nuclear Research (CERN, Geneva) in 1995. She was a regular visiting scholar of the Experimental Software Engineering Group at the University of Maryland from 1998 to 2000, visiting scientist at the Fraunhofer Institute for Experimental Software Engineering in Germany in 2002 and regular visiting professor at the M3S group of the University of Oulu in Finland from 2015 to 2017.

Affiliations

Iflaah Salman[1] · **Burak Turhan**[2] · **Sira Vegas**[3]

Burak Turhan
burak.turhan@monash.edu

Sira Vegas
svegas@fi.upm.es

[1] M3S Group, University of Oulu, Oulu, Finland

[2] Faculty of Information Technology, Monash University, VIC, Australia

[3] Escuela Tecnica Superior de Ingenieros Informaticos, Universidad Politecnica de Madrid, Madrid, Spain