On the Effect of Noise on Software Engineers' Performance: Results from Two Replicated Experiments

Simone Romano
University of Bari
Bari, Italy
simone.romano@uniba.it

Giuseppe Scanniello
University of Basilicata
Potenza, Italy
giuseppe.scanniello@unibas.it

Maria Teresa Baldassarre
University of Bari
Bari, Italy
mariateresa.baldassarre@uniba.it

Davide Fucci
Blekinge Institute of Technology
Karlskrona, Sweden
davide.fucci@bth.se

***Abstract*—Noise, defined as an unwanted sound, is one of the most common factors people have to deal with when performing their daily working tasks. Researchers have marginally investigated the effect of noise on software engineers' performance. In this paper, we present the results of two replicated experiments whose main goal was to increase the body of knowledge, by confirming or not the results of the baseline experiments, on the effect of noise while comprehending functional requirements specifications and fixing faults in source code. The results of the replicated experiments suggest that: *(i)* noise does not significantly affect the comprehension of functional requirements specifications and *(ii)* noise significantly and negatively affects fixing faults if this task lasts 30 minutes, while it does not have a significant impact if the task lasts 60 minutes. The results of the replications confirm to a large extent those of the baseline experiments and allow us to postulate, as done for the baseline experiments, that fixing faults is more vulnerable to noise than comprehending the specifications of functional requirements.**

***Index Terms*—Noise, Comprehension of Functional Requirements Specifications, Fault Fixing, Experiment, Replication.**

I. Introduction

Software Engineering (SE) requires the cycle of model building, experimentation, and learning. Experimentation is fundamental to any scientific research field, including SE [1], where the researcher needs laboratories to study problems (*e.g.,* from the field faced by practitioners) and develop and evolve solutions based on experimentation and empirical evidence. To this end, a number of empirical approaches are possible (*e.g.,* experiments, case studies, and so on). Experiments are important to test hypotheses and, in particular, the predictive ability of hypotheses. If new experiments support certain hypotheses, then we have more evidence in favor of these hypotheses [2]. As evidence grows and becomes stronger, the hypotheses can be accepted as a scientific theory and transferred to other contexts [3]. A possible tool to pursue this goal is to conduct replications of baseline experiments. Replications of experiments involve repeating the investigation under similar conditions while, for example, varying the subject population [4].

When experimenting, both researcher and practitioner need to understand the nature of SE discipline. In SE, the technologies are mostly human-intensive, rather than automated. Like manufacturing, the major problem is understanding and improving the relationship between the processes and the products they create. But unlike manufacturing, the process in SE is development, not production [1]. Development (design, implementation, and maintenance phases) takes place in workspaces that, nowadays, tend to have less privacy, with less dedicated space, which leads to noisy environments. Software companies that provide a noisy workplace are comforted by the belief that this factor does not matter [5], but noise exerts its specific influences on various forms of cognitive responses [6]. After all, software engineers are knowledge workers—they need to have their brain in gear to do their work—and thus their performance would be sensitive to a noisy workplace.

The effect of noise on SE tasks has been marginally studied in the past (*e.g.,* [7]–[9]). Romano *et al.* [9] conducted two controlled experiments,[1] whose primary goal was, respectively, to study whether noise affects software engineers' performance in the following two kinds of tasks/processes: *(i)* comprehending functional requirements specifications and *(ii)* fixing faults (or bugs from here on) in source code. The results suggest that software engineers have significantly worse performance in fixing faults in source code when exposed to noise, while no statistically significant difference is present while comprehending functional requirements. The authors conjectured that bug fixing is more vulnerable to noise than comprehending functional requirements because bug fixing seems to be a more resource-demanding task.

To increase our confidence in the results of the baseline experiments by Romano *et al.* [9], we replicated these experiments with more experienced participants (undergraduate vs. graduate students) from the same university. Our goal, when designing the replications, was to run them as exact as possible to the baseline experiments. This was to be confident that, in case of differences in the results of baseline and replication experiments, such a difference is mostly due to the participants' experience.

Paper structure. In Section II, we discuss related work and

[1]One of these experiments was preliminary presented in a poster paper [8].

978-1-7281-9532-2/20/$31.00 ©2020 IEEE
DOI 10.1109/SEAA51224.2020.00062

present background information while, in Section III, we summarize the baseline experiments by Romano *et al.* [9]. In Section IV, we present the replicated experiments as well as the obtained results. In Section V, we discuss the results and how they increased our body of knowledge by combining outcomes from the baseline and replicated experiments. In Section VI, we discuss potential threats to the validity of our results. Final remarks conclude the paper.

II. Related Work and Background

In this section, we review the literature related to our study and then summarize the theories concerning the effect of noise on individual performance.

A. Related Work

There is a series of empirical investigations aimed to understand how a mental state of software developers impacts their daily programming activities. *Flow* is a mental state of high concentration that results in an absolute assimilation in the activity at hand (*e.g.,* software development) [10]. Disturbance from the surrounding environment (*e.g.,* due to noise) can disrupt causing work fragmentation and negative impact on productivity [11]. For example, Meyer *et al.* [12] found that developers feel productive when they are not interrupted and they do not need to switch between different tasks.

Researchers have long recognized the detrimental effects of disturbance in the developers' workplace. In this respect, the SE and HCI (Human Computer Interaction) communities have proposed different solutions to the problems related to developers' interruptibility while accomplishing a task [13]–[15]. Gievska *et al.* [13] proposed an interruptibility model to mediate human interruptions by a computer and a study with 24 knowledge workers (not software developers). The authors observed, through the application of their model, that reducing interruptions can increase the perceived quality of work while decreasing frustration. In the context of software developers, Iqbal and Bailey [14] proposed a system that would postpone possible causes of disturbance to a more apt time based on cognitive theory. These authors also conducted an empirical study with six professionals, whose results indicated that the proposed approach reduces frustrations while yielding to faster reaction time. More recently, Züger *et al.* [15] developed a physical device that would signal to the surrounding environment (*e.g.,* co-workers) the best moment to disturb (or not) a developer. The results of a field study showed increased awareness about the disrupting effects of such kind of disturbance.

DeMarco and Lister related noise and other environmental factors (*e.g.,* space) to software developers' performance [7]. In a study with 166 professionals, they observed that a quiet and commodious workplace could improve productivity (*e.g.,* time to complete the task) by a factor of 2.6. The main differences with respect to our study can be summarized as follows: *(i)* we conducted two experiments in controlled conditions on two kinds of SE tasks and *(ii)* we quantitatively assessed the effect of noise on the comprehension of functional requirements and the capability to fix faults in source code.

B. Background

A summary of the reference theories regarding the effect of noise on individuals' performances follows.

Arousal Theory. Broadbent [16], in his theory, invoked an arousal induced attentional narrowing mechanism: noise increases arousal of an individual, which decreases his/her breadth of attention. At a lower level of arousal, an individual tends to exclude task-irrelevant cues, and thus the attentional narrowing facilitates performance. However, beyond a certain arousal "optimal" level, performance is impaired because its increase in arousal might cause increased narrowing so that task-relevant cues are excluded. Broadbent postulates that more demanding tasks should have lower levels of optimum arousal. These tasks should yield the greatest performance decrements in the presence of noise. This is to say that cognitive tasks suffer greater magnitudes of performance impairment with respect to less demanding tasks. In addition, noise intensity and duration influence the arousal levels. In other words, a higher intensity and longer duration of noise cause greater negative effects on performance. As for schedule, intermittent noise should impair performance more than continuous one. To summarize, noise effect varies according to the kind of task and the noise intensity, duration, and schedule.

Composite Theory. Poulton's theory [17] predicts that noise effects should degrade performance for those conditions in which inner speech[2] is masked. The gains in performance in continuous noise, early in the task, occur because the increase in arousal compensates for the detrimental effects of masking. However, with time on task, the arousal decreases and the masking effect dominates. To summarize, the noise effect, in the composite theory, is considered similar across task and kind of noise, but a moderating effect is expected for intensity, duration, and schedule.

Maximal Adaptability Theory. This theory was proposed by Hancock and Warm [19]. The authors postulate that stress, and noise is a source of stress, can be accounted for in three loci. *Input* represents objective environmental and task factors, *adaptation* concerns the capability of an individual to cope with demands intrinsic to an environment, and *output* refers to the response about the task environment. The output of a task depends on the characteristics of an individual and it might be affected by noise. As for adaptation, noise can impair the capacity through the masking or distortion of task-relevant auditory information. According to the maximal adaptability theory, there is a threshold of dynamic instability in which the adaptation of the individual fails and followed by a decrease in performance. Hancock and Warm assert that the performance on more resource-demanding cognitive tasks, in case of noise, is more impaired than performance on motor or perceptual

[2]Also referred to as verbal thinking, inner speaking, covert self-talk, internal monologue, and internal dialog. Inner speech is thinking in words and also refers to the semi-constant internal monologue some individuals have with themselves at either conscious or semi-conscious level [18].

tasks. In addition, for higher noise intensity and longer noise duration, a greater performance impairment is present. In cognitive tasks, speech noise is more disruptive than non-speech. Also, noise schedule could affect performance. To summarize, the maximal adaptability theory indicates that noise effect varies as a function of the task and noise kind, as well as schedule, duration, and intensity.

Summary. Arousal, composite, and maximal adaptability theories predict similar results on noise effects for certain variables (*e.g.,* noise intensity, duration, and schedule), but different results for others (*e.g.,* kind of task and noise). Szalma and Hancock [6] have recently conducted a meta-analysis of noise effects on individual performance. Results confirm the predictions of these three theories only partially. In particular, Szalma and Hancock observed that noise effects varied as a function of the kind of noise and task, and noise intensity, duration, and schedule. Results also indicated that shorter duration of noise has greater detrimental effects on performance than longer duration.

III. Baseline Experiments

In this section, we report the baseline experiments by Romano *et al.* [9] by taking into account guidelines [20] for reporting replicated experiments.[3]

A. Research Questions

Romano *et al.* planned the baseline experiments to study the following Research Questions (RQs):

- **RQ1.** Does noise worsen software engineers' performance in comprehending functional requirements specifications?
- **RQ2.** Does noise worsen software engineers' performance in fixing faults in source code?

B. Participants, Artifacts, and Tasks

The participants were recruited among the last-year undergraduate students, who were taking a SE course, in the Computer Science program, at the University of Basilicata (Italy). The participants voluntarily took part in the experiments—*i.e.,* they were neither paid nor obliged to participate. Among the students taking the SE course, 55 took part in the first experiment on the comprehension of functional requirements specifications (*i.e.,* Exp1_c from here on), while 42 also took part in the second experiment on fixing bugs in source code (*i.e.,* Exp1_f from here on).

The experiments represented two optional didactic activities of the SE course. This course covered the following topics: software development processes, requirements specification, software design, maintenance, and testing. Throughout the course, participants also practiced both modeling of functional requirements and bug fixing. This is because they carried out a number of homework and classwork assignments on these kinds of SE tasks. The participants were also experienced with Java programming.

[3]Despite our effort to report as much information as possible about the baselines experiments, we could not report here some information for space reasons—missing information can be found in the paper by Romano *et al.* [9].

In Exp1_c, the participants were asked to perform tasks of comprehension of functional requirements specifications on the following two systems:

- **M-Shop**, a system for managing the sales of a music shop.
- **Theater**, a system for managing the ticket reservation of a theater.

To comprehend functional requirements specifications, each participant received the functional, object, and dynamic models associated with the requirements understand as well as a comprehension questionnaire consisting of 11 closed-ended questions—each question admitted one or more right answer. Both systems, as well as the considered functional requirements, were part of the experimental material Abrahão *et al.* made available [21]—*i.e.,* Romano *et al.* reused that experimental material in Exp1_c.

In Exp1_f, the participants had to perform fault fixing tasks on the following two systems:

- **LaTazza**, a Java desktop application for managing sales and supplies of beverages for a coffee maker.
- **AveCalc**, a Java desktop application for managing the exams of a university student.

To perform each fault fixing task, the participants received the codebase (without test cases) and mission of the buggy system, along with six bug reports—*i.e.,* one bug report for each fault the participants had to fix in that codebase. Similarly to Exp1_c, Romano *et al.* reused in Exp1_f the experimental material that past studies made available [22], [23].

The duration of each comprehension task was fixed at 30 minutes, while that of each fault fixing task was set at 60 minutes. Such duration for the experimental tasks was based on results shown in past studies [21], [23].

C. Variables and Hypotheses

The main independent variable (manipulated in both Exp1_c and Exp1_f) was **Condition**. This variable assumed two values, namely: *NOISE—i.e.,* participants performing the experimental tasks (either comprehending functional requirements specifications or fixing faults) exposed to noise—and *NORMAL—i.e.,* participants performing the experimental tasks in a normal condition.

In Exp1_c, Romano *et al.* used two Dependent Variables (DVs), namely $\boldsymbol{F_c}$ and $\boldsymbol{Avg}$, to measure software engineers' performance in comprehending functional requirements specifications. F_c is the (balanced) *F-measure* [24] of *precision* (correctness) and *recall* (completeness) of the answers a certain participant gave to the comprehension questionnaire. Formally, F_c is defined as follows:

$$F_c = \frac{2 * P_c * R_c}{P_c + R_c}$$

P_c and R_c are precision and recall, respectively computed as:

$$P_c = \frac{\sum_{i=1}^{11} |answ_i \cap oracle_i|}{|\sum_{i=1}^{11} answ_i|} R_c = \frac{\sum_{i=1}^{11} |answ_i \cap oracle_i|}{|\sum_{i=1}^{11} oracle_i|}$$

where $answ_i$ is the set of answers the participant provided for the question i, $oracle_i$ is the set of correct expected answers

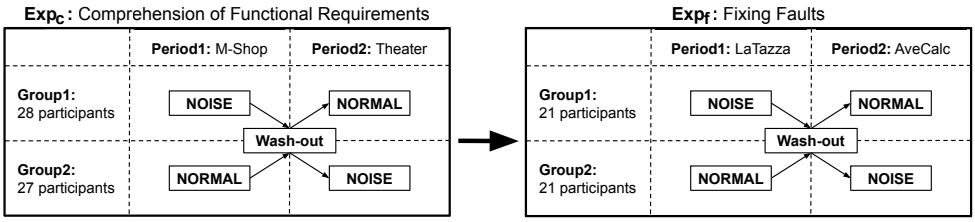


Fig. 1: Overall design of the baseline experiments.

(*i.e.,* the oracle) for the question i, and 11 is the number of questions. F_c is a trade-off measure that equally weights correctness and completeness.

As for Avg, it was computed as follows:

$$Avg = \frac{\sum_{i=1}^{11} count_i}{11}$$

where $count_i$ is equal to: 1 if the set of answers provided by the participant to the question i corresponded to the oracle for the question; 0 otherwise. F_c and Avg range in $[0, 1]$—for both DVs, the greater the value, the better the comprehension of the specifications of functional requirements is. The use of F_c and Avg to measure the comprehension of functional requirements specifications was ground on past work [21], [25].

In Exp1$_f$, $\boldsymbol{F_f}$ was used to measure software engineers' performance in fixing faults. F_f is the (balanced) *F-measure* [24] of the *precision* (correctness) and *recall* (completeness) of the faults a participant fixed. F_f is defined as follows:

$$F_f = \frac{2 * P_f * R_f}{P_f + R_f}$$

P_f and R_f are precision and recall, respectively computed as:

$$P_f = \frac{\#bugsCorrectlyFixed}{\#bugsFixed} R_f = \frac{\#bugsCorrectlyFixed}{6}$$

where $\#bugsFixed$ is the count of faults the participant had fixed both correctly and incorrectly, while $\#bugsCorrectlyFixed$ is the count of faults the participant had fixed correctly. To determine that count, Romano *et al.* used the acceptance test suites provided by Scanniello *et al.* [23]. F_f ranges in $[0, 1]$—the greater the value, the better the performance in fixing faults is. The use of F_f to measure the performance in fixing faults was ground on prior work [23].

Two null hypotheses, one for each RQ, were formulated:

- **Hn1.** Noise does not significantly affect software engineers' performance in comprehending functional requirements specifications.
- **Hn2.** Noise does not significantly affect software engineers' performance in fixing faults.

In case a null hypothesis is rejected, it is possible to accept the alternative one. For example, the alternative hypothesis for Hn1 is: noise significantly affects software engineers' performance in comprehending functional requirements.

D. Design and Execution

Figure 1 shows how the baseline experiments were arranged. For both Exp1$_c$ and Exp1$_f$, the experimental design was *AB/BA* [26]. The participants were randomly split into two experimental groups (*i.e.,* Group1 and Group2) and received both treatments (*i.e.,* NOISE and NORMAL) once in each experiment. Group1 received the NOISE treatment in the first period[4] and the NORMAL treatment in the second one, while Group2 the opposite. In an AB/BA design, the experimental group defines the sequence with which the treatments are applied in the first and second periods. Regardless of the treatment, the participants dealt with the same system in the same period—*e.g.,* any participant in Exp1$_c$ tackled M-Shop in the first period and Theater in the second. Within each experiment, there was a 30-minute *wash-out* period—*i.e.,* a period to leave enough time for the effect of a treatment to recede completely before applying another treatment [26]—between the first and second periods. *Wash-out* periods are used in *AB/BA* designs to counteract the *carryover effect* [26], which is an internal validity threat [1]. The two experiments were planned in two different days (*i.e.,*Exp1$_c$ first and then Exp1$_f$).

The participants working in a noise condition performed the experimental tasks (in both Exp1$_c$ and Exp1$_f$) with a noise exposure level[5] (L_{EX}) equal to 82dB. Such a L_{EX} value was established based on the Directive 2003/10/EC2 of the European Parliament.[6] Romano *et al.* chose a L_{EX} value equal to 82dB because: *(i)* it does not require the use of individual hearing protectors; and *(ii)* it was close but inferior to the limit of 85dB for which workers shall wear individual hearing protectors because deemed harmful to health. The kind of noise was speech as it is the major type of practical distractive noise [6] and it is common in workplaces with open offices [27]. For the participants working in normal conditions, the LEX value (*i.e.,* 42dB) was that of a quiet office workplace [27].

The NOISE treatment was always administered to the participants in Lab1, a research laboratory equipped with ceiling speakers. As for the NORMAL treatment, it was always administered in another research laboratory, Lab2, far from sources of noise (*e.g.,* road traffic).

E. Data Analysis and Results

As suggested by Wellek and Blettner [28], Romano *et al.* ran the pre-test to check the assumption of negligible carryover effect, for each DV, before testing the corresponding null

[4]A period is defined as the time at which a treatment is applied [26].

[5]It is defined as the time-weighted average of the noise exposure levels.

[6]https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02003L0010-20081211

TABLE I: Results from statistical inference for the baseline experiments.

Experiment	Variable	Carryover Effect	Condition Effect	Hypotheses Testing Outcome
Exp1_c	F_c	0.518 (t-test)	0.731 (t-test)	Failed to reject Hn1
	Avg	0.431 (t-test)	0.312 (Mann-Whitney U test)	Failed to reject Hn1
Exp1_f	F_f	0.036 (t-test)	0.024 (Mann-Whitney U test)	Hn2 rejected in favor of NORMAL

hypothesis. The pre-test consists of an unpaired two-sided t-test (or an unpaired two-sided Mann-Whitney U test if the normality assumption[7] of the t-test is not met) to compare the *within-participant sums* of the DV values (in both periods) for the two experimental groups [28]. As shown in Table I (see Carryover Effect), the p-values returned by the two pre-tests (one for F_c and one for Avg) for Exp1_c were both not less than the significance level (α) fixed at 0.05. This is to say that the carryover effect was negligible for both F_c and Avg. Such an outcome allowed Romano *et al.* to test, as suggested by Wellek and Blettner [28], the corresponding null hypothesis (*i.e.,* Hn1) for both F_c and Avg through the unpaired two-sided t-test (or an unpaired two-sided Mann-Whitney U test if the normality assumption of the t-test was not met). This test was used to compare the *within-participant differences* of the DV values (in both periods) for the two experimental groups [28]. The p-values for Hn1 (see Condition Effect in Table I) were both not less than α, thus Hn1 could be rejected for neither F_c nor Avg. Accordingly, Romano *et al.* concluded that: *noise does not significantly affect the comprehension of the specifications of functional requirements*.

As for Exp1_f, the pre-test indicated that the carryover effect was not negligible on F_f—the p-value was less than α (see Carryover Effect in Table I). As suggested in the literature [26], Romano *et al.* discarded the second period for F_f and tested the null hypothesis (*i.e.,* Hn2) by taking into account the first period only. To that end, Romano *et al.* planned to run an unpaired two-sided t-test (or an unpaired two-sided Mann-Whitney U test if the normality assumption of the t-test was not met) comparing the F_f values of the NOISE and NORMAL conditions. The p-value was less than α (see Condition Effect in Table I) so allowing rejecting Hn2. The difference between the NOISE and NORMAL conditions was in favor of the NORMAL condition. Accordingly, Romano *et al.* concluded that: *noise significantly and negatively affect the fixing of faults in source code*.

Summing up, the results from Exp1_c and Exp1_f suggest that, while noise does not significantly affect the comprehension of the specifications of functional requirements, it has a significant and negative impact on the fixing of faults in source code. To determine whether the duration of noise exposure (30 minutes in Exp1_c vs. 60 minutes in Exp1_f) could explain the above-mentioned outcomes, Romano *et al.* analyzed the first 30 minutes of Exp1_f (first period). In particular, they ran an unpaired two-sided Mann-Whitney U test (as the normality assumption of the t-test was not met) comparing the F_f values, computed on the first 30 minutes, of the NOISE and NORMAL conditions. The returned p-value (0.028) indicated a significant difference, which was in favor of the NORMAL condition. Accordingly, Romano *et al.* excluded that the duration of noise exposure was behind the lack of significant difference in Exp1_c. They then conjectured, in line with the literature [6], [16], [19], that: *bug fixing is more vulnerable to noise than comprehending the specifications of functional requirements*. This is why the results did not indicate any significant difference in Exp1_c.

[7]To test the normality assumption, the Shapiro-Wilk test was used.

IV. Replicated Experiments

The SE community has been embracing replications more readily (*e.g.,* [4], [29]–[31]). Despite this important trend, there is no agreement yet on terminology, typology, purposes, operation, and other replication issues [29], [30], [32]. In general, we can define a replication as the repetition of an experiment [33]. There are two primary motivations to perform replications: *(i)* they are necessary to solve problems and to collect evidence in order to bring credibility to a given study and *(ii)* they are valuable because the obtained evidence can be used in the daily activities of practitioners [30], [32].

In the rest of this section, we present our replications (from here onwards referred to as Exp2_c and Exp2_f, respectively) and the obtained results. For space reasons, the replications are described in terms of differences with respect to the baseline experiments. In particular, we did not describe RQs, artifacts, tasks, variables, hypotheses, design, and data analysis of the replications because these details are the same as the baseline experiments and can be found in Section III.

A. Participants

The participants were last-year graduate students in Computer Engineering at the University of Basilicata (Italy) who were taking the Advanced SE course. The participation in the study was voluntary—*i.e.,* they were neither paid nor obliged to participate. Among the 16 students taking the course, 15 decided to take part in the replicated experiments. Any participant took part in both Exp2_c and Exp2_f. The participants were experienced with C, C++, and Java programming. Based on their curricula, the participants had passed the following exams about software design and programming: SE, Procedural Programming, Object-Oriented Programming, Advanced Object-Oriented Programming, and Mobile Programming. They also had knowledge of UML-based software modeling and were experienced with fixing bugs in source code written by other developers [34].

B. Execution

We executed the replicated experiments as much close as possible to the baseline ones. This is to say that we used an AB/BA design for any experiment and randomly assigned

TABLE II: Results from statistical inference for the replicated experiments.

Experiment	Variable	Carryover Effect	Condition Effect	Hypotheses Testing Outcome
$Exp2_c$	F_c	0.41 (t-test)	0.672 (t-test)	Failed to reject Hn1
	Avg	1 (Mann-Whitney test)	0.436 (t-test)	Failed to reject Hn1
$Exp2_f$	F_f	0.256 (t-test)	0.163 (t-test)	Failed to reject Hn2

TABLE III: Some descriptive statistics for the replicated experiments and baseline ones.

Variable	Statistic	Replicated Experiments		Baseline experiments	
		NORMAL	NOISE	NORMAL	NOISE
F_c	Median	0.8	0.75	0.692	0.714
	Mean	0.78	0.764	0.678	0.683
	SD	0.096	0.097	0.127	0.14
Avg	Median	0.545	0.545	0.545	0.545
	Mean	0.576	0.545	0.502	0.526
	SD	0.089	0.103	0.13	0.155
F_f	Median	0.667	0.545	0.5	0.286
	Mean	0.67	0.518	0.481	0.315
	SD	0.261	0.349	0.297	0.198

eight participants to Group1 and seven to Group2. In the replications, we reproduced an environment to accomplish the experimental tasks as similar as possible to that of the baseline experiments. The laboratory used to accomplish the experimental tasks in a noise condition was equipped with ceiling speakers to reproduce the same speech noise as the baseline experiments. We used a phonometer to be sure that the participants were exposed to the same established noise level (*i.e.,* $L_{EX} = 82$dB). To administer the NORMAL treatment, we simulated a quite office workplace and used a phonometer to regularly measure the noise level—the L_{EX} value was about 42dB.

C. Results

The raw data of our replications on available on the web.[8] To analyze these data, we applied the same analysis procedure as the baseline experiments. The results indicated that the carryover effect was negligible in both the replications (p-values are not less than α) whatever the DV was (see Carryover Effect in Table II).

In Table III, we report median, mean, and SD (Standard Deviation) of the values of the DVs for each replicated experiment (and baseline one) and condition. As for $Exp2_c$, the distributions of the F_c values for the NORMAL and NOISE conditions seem similar (see also the boxplots shown in Figure 2a). Similar considerations can be done for Avg (see Table III and Figure 2a). Results from hypotheses testing do not allow us to reject Hn1 for both F_c and Avg as suggested by the p-values reported in Table III. On the basis of the observed results, we can answer RQ1 as follows: *noise does not significantly affect the comprehension of the specifications of functional requirements*.

As for $Exp2_f$, we can observe some differences in the descriptive statistics of the NORMAL and NOISE conditions in favor of NORMAL (see Table III and Figure 2b). However, the p-value returned by the t-test does not allow rejecting Hn2. Similarly to $Exp1_f$, we also analyzed the F_f values by taking into account the first 30 minutes only. In particular, we ran the pre-test (*i.e.,* an unpaired two-sided t-test comparing the within-participant sums of the F_f values for the two experimental groups), which indicated a negligible carryover effect (p-value $= 0.534$). Then, we ran an unpaired two-sided t-test comparing the within-participant differences of the F_f values for the two experimental groups. The returned p-value was equal to 0.041, thus Hn2 could be rejected in favor of NORMAL. The results from $Exp2_f$ allow us to answer RQ2 as follows: *noise negatively and significantly affects the fixing of faults in source code when the duration of noise exposure is 30 minutes, while it does not have a significant impact when the duration of noise exposure is 30 minutes*.

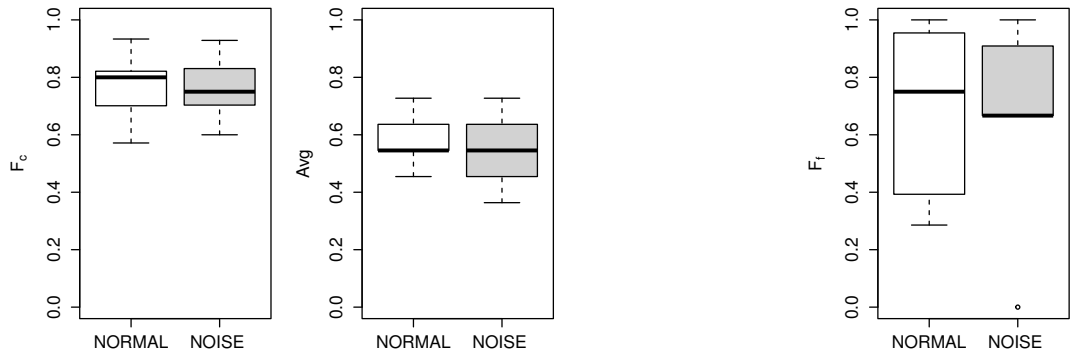
Finally, the descriptive statistics for the replications as compared to those of the baseline experiments (see Table III) seem to indicate that the performance of the participants in the replications was better or, in the worst case, similar to that of the participants in the baseline experiments whatever the DV and condition are. This confirms that the participants in the replications were more experienced than those in the baseline experiments.

V. Discussion

Replications are successful when they help the research community to build knowledge about which outcomes (or observations) hold under which conditions. Therefore, replications that produce the same outcomes as the baseline experiments are as useful, for the research community, as replications that fail to produce the same outcomes as the baseline experiments [30]. $Exp2_c$ falls in the former case. In particular, $Exp2_c$ allows confirming that noise does not significantly affect the comprehension of the specifications of functional requirements. Such evidence has a practical implication and seems to corroborate the choice, made by some software companies, of providing noisy workspaces to developers. This is because these software companies deem that this factor does not matter and thus they can save money on workspaces [35].

As far as $Exp2_f$ is concerned, we were not able to fully support the results from $Exp1_f$. In particular, the results from $Exp2_f$ suggest that the effect of noise on fixing bugs is not significant when the noise exposure is 60 minutes, while the results from $Exp1_f$ suggest a significant effect. We can speculate that the higher the experience of developers, the less the negative impact of noise on fault fixing. In fact, participants in $Exp2_f$ were last-year graduate students, while

[8]https://doi.org/10.6084/m9.figshare.12504944.v1



(a) (b)

Fig. 2: Boxplots, by Condition, for (a) F_c and Avg in Exp2$_c$, and (b) F_f in Exp2$_f$.

those in Exp1$_f$ were last-year undergraduate students. This outcome is clearly relevant for the researcher who could be interested in further study the relation between developers' experience and duration of noise exposure. This outcome is also relevant for the practitioner because a software company should prevent to expose developers with a low experience to noise when accomplishing bug fixing.

Based on the results of Exp1$_c$ and Exp1$_f$, Romano *et al.* conjectured that bug fixing is more vulnerable to noise than comprehending the specifications of functional requirements. Our results seem to support the above-mentioned conjecture since, under the same duration of 30-minute noise exposure, noise affects the fixing of bugs while it does not affect the comprehension of the specifications of functional requirements. This outcome is of interest for those researchers who want to study which SE tasks are more vulnerable to noise.

Finally, the results of bug fixing on 30-minute and 60-minutes noise exposure suggest that the more the experience of developers, the better developers counteract the negative effect of noise, which initially affects their performance in fixing bugs. This outcome is important for both the researcher who could be interested in investigating the detrimental effect of noise duration on developers' performance when accomplishing SE tasks and fault fixing, in particular.

VI. Threats to Validity

In this section, we discuss the threats that could affect the validity of our results. Since we replicated the baseline experiments as closely as possible, our replications inherit most threats of the baseline experiments, although we mitigated some threats with respect the baseline experiments (*e.g.,* threats to external validity). We present these threats based on the guidelines by Wohlin *et al.* [1].

Internal Validity. A *maturation* threat might exist because the participants exposed to noise might be more motivated to accomplish the experimental tasks (*e.g.,* due to arousal the effect) [16], [19]. We mitigated a threat of *diffusion or imitation of treatments* in several way: *(i)* we monitored the participants during the execution of each task to prevent that they exchanged information on the tasks; *(ii)* we took back, at the end of each experimental task, any experimental material we gave the participants; and *(iii)* the participants in Group1 and Group2 performed the same task at the same time in each replication. The *selection* threat of letting volunteers take part in the experiments could influence the results.

Construct Validity. We exploited metrics well-known and widely-adopted to measure the constructs (*e.g.,* [21], [23]). However, we considered only one metric to assess participants' performances (*i.e., mono-method bias*) in Exp2$_f$. The participants were not informed about our research goal; however, they could be aware of being part of an experiment on noise effect. Thus, there could be the risk of *hypotheses guessing*.

Conclusion Validity. The use of an AB/BA design might affect results due to carryover effect. To dealt with this kind of threat, we introduce a wash-out period in each replication and studied, similarly to past studies [9], [36], if this period was long enough to neutralize the carryover effect. The implementation of a treatment (*e.g.,* NOISE) might be not similar between different participants and among the experiments (*i.e., reliability of treatment implementation*). We mitigate this kind of threat in the replications by implementing NOISE and NORMAL treatments as standard as possible over different participants. Finally, there might be a threat to conclusion validity due to the number of participants.

External Validity. Working with students poses a threat of *interaction of selection and treatment*. However, working with students also implies various advantages, such as their homogeneous prior knowledge [37]. Since the participants in the replications were more experienced than those in the baseline experiments we mitigated this kind of threat to external validity. Nevertheless, we foster replications with professional developers (*e.g.,* from industry). The use of UML as modeling notation in Exp2$_f$ might affect results (*i.e., interaction of setting and treatment*). However, UML is a de-facto standard for software modeling and the participants were familiar with such notation. Another threat of interaction of setting and treatment is related to the used experimental tasks that could not be representative of real-world tasks. However, the tasks we asked the participants to accomplish in Exp2$_c$ and Exp2$_f$ were used in past empirical studies [21], [23]. Also, the experimental tasks should equally affect the results of the participants when exposed or not to noise in both Exp2$_c$ and Exp2$_f$.

VII. Conclusion

In this paper, we present the results of two replicated experiments on the effect of noise while comprehending functional requirements specifications and fixing faults in source code. The results suggest that noise: *(i)* does not significantly affect the comprehension of functional requirements specifications and *(ii)* significantly affects fault fixing if this task lasts less than 30 minutes while it does not have a significant impact if the task lasts 60 minutes. The results of the replications confirm to a large extent those of the baseline experiments so increasing our body of knowledge on the effect of noise on some kinds of SE tasks. In particular, the results from our replications and those form the baseline experiments suggest that bug fixing is more vulnerable to noise than comprehending the specifications of functional requirements. To confirm or not these findings, further empirical investigations are needed.

Acknowledgement

This work was partially supported by: the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology; the "Digital Service Ecosystem" project (Cod. PON03PE-00136-1) funded by Italian MIUR; and "Auriga2020" project (Cod. T5LXK18) funded by Apulia Region.

References

[1] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[2] N. Juristo and A. Moreno, *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, 2001.

[3] P. Ardimento, D. Caivano, M. Cimitile, and G. Visaggio, "Empirical investigation of the efficacy and efficiency of tools for transferring software engineering knowledge," *Journal of Information and Knowledge Management*, vol. 7, no. 3, pp. 197–207, 2008.

[4] J. C. Carver, N. Juristo, M. T. Baldassarre, and S. Vegas, "Replications of software engineering experiments," *Empir. Softw. Eng.*, vol. 19, no. 2, pp. 267–276, Apr 2014.

[5] T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams*, 3rd ed. Addison-Wesley Professional, 2013.

[6] J. L. Szalma and P. A. Hancock, "Noise effects on human performance: a meta-analytic synthesis," *Psychological bulletin*, vol. 137, no. 4, pp. 682–707, 2011.

[7] T. DeMarco and T. Lister, "Programmer performance and the effects of the workplace," in *Proc. of the International Conference on Software Engineering*. IEEE, 1985, pp. 268–272.

[8] S. Romano, G. Scanniello, D. Fucci, N. Juristo, and B. Turhan, "Poster: The effect of noise on requirements comprehension," in *Proc. of International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 308–309.

[9] S. Romano, G. Scanniello, D. Fucci, N. Juristo, and B. Turhan, "The effect of noise on software engineers' performance," in *Proc. of International Symposium on Empirical Software Engineering and Measurement*. ACM, 2018.

[10] M. Csikszentmihalyi, *Creativity: Flow and the Psychology of Discovery and Invention (Harper Perennial Modern Classics)*. HarperCollins e-books, 2009.

[11] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz, "The Work Life of Developers: Activities, Switches and Perceived Productivity," *IEEE Trans. on Softw. Eng.*, vol. PP, no. 99, pp. 1–1, 2017.

[12] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, "Software developers' perceptions of productivity," in *Proc. of International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 19–29.

[13] S. Gievska, R. Lindeman, and J. Sibert, "Examining the qualitative gains of mediating human interruptions during hci," in *Proc. of the International Conference on Human-Computer Interaction*, 2005.

[14] S. T. Iqbal and B. P. Bailey, "Understanding and developing models for detecting and differentiating breakpoints during interactive tasks," in *Proc. of SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2007, pp. 697–706.

[15] M. Züger, C. Corley, A. N. Meyer, B. Li, T. Fritz, D. Shepherd, V. Augustine, P. Francis, N. Kraft, and W. Snipes, "Reducing interruptions at work: A large-scale field study of flowlight," in *Proc. of CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 61–72.

[16] D. Broadbent, "The current state of noise research: Reply to poulton," *Psychological Bulletin*, vol. 85, pp. 1052–1067, 1978.

[17] E. Poulton, "Composite model for human performance in continuous noise," *Psychological Review*, vol. 86, pp. 361–375, 1979.

[18] A. Morin, J. D. Runyan, and T. M. Brinthaupt, "Editorial: Inner experiences: Theory, measurement, frequency, content, and functions," *Frontiers in Psychology*, vol. 6, p. 1758, 2015.

[19] P. A. Hancock and J. S. Warm, "A dynamic model of stress and sustained attention," *Human Factors*, vol. 31, no. 5, pp. 519–537, 1989.

[20] M. T. Baldassarre, J. Carver, O. Dieste, and N. Juristo, "Replication types: towards a shared taxonomy," in *Proc. of International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014, pp. 18:1–18:4.

[21] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, "Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments," *IEEE Trans. on Softw. Eng.*, vol. 39, no. 3, pp. 327–342, 2013.

[22] F. Ricca, M. D. Penta, M. Torchiano, P. Tonella, M. Ceccato, and C. A. Visaggio, "Are fit tables really talking?" in *Proc. of International Conference on Software Engineering*. IEEE, 2008, pp. 361–370.

[23] G. Scanniello, M. Risi, P. Tramontana, and S. Romano, "Fixing faults in c and java source code: Abbreviated vs. full-word identifier names," *ACM Trans. Softw. Eng. Methodol.*, vol. 26, no. 2, pp. 6:1–6:43, 2017.

[24] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[25] E. Kamsties, A. von Knethen, and R. Reussner, "A controlled experiment to evaluate how styles affect the understandability of requirements specifications." *Inf. Softw. Technol.*, vol. 45, no. 14, pp. 955–965, 2003.

[26] S. Vegas, C. Apa, and N. Juristo, "Crossover designs in software engineering experiments: Benefits and perils," *IEEE Trans. on Softw. Eng.*, vol. 42, no. 2, pp. 120–135, 2016.

[27] L. E. Maxwell, "Noise in the office workplace," *Facility Planning and Management Notes*, vol. 1, no. 11, 2015.

[28] S. Wellek and M. Blettner, "On the Proper Use of the Crossover Design in Clinical Trials," *Dtsch Arztebl International*, vol. 109, no. 15, pp. 276–281, 2012.

[29] B. Kitchenham, "The role of replications in empirical software engineering - a word of warning," *Empir. Softw. Eng.*, vol. 13, no. 2, pp. 219–221, 2008.

[30] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical software engineering," *Empir. Softw. Eng.*, vol. 13, no. 2, pp. 211–218, 2008.

[31] F. Q. B. da Silva, M. Suassuna, A. C. C. França, A. M. Grubb, T. B. Gouveia, C. V. F. Monteiro, and I. E. dos Santos, "Replication of empirical studies in software engineering research: a systematic mapping study," *Empir. Softw. Eng.*, vol. 19, no. 3, pp. 501–557, 2014.

[32] O. S. Gómez, N. J. Juzgado, and S. Vegas, "Understanding replication of experiments in software engineering: A classification," *Information & Software Technology*, vol. 56, no. 8, pp. 1033–1048, 2014.

[33] V. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *IEEE Trans. on Soft. Eng.*, vol. 25, no. 4, pp. 456–473, 1999.

[34] A. Fernández-Sáez, M. Genero, D. Caivano, and M. Chaudron, "Does the level of detail of uml diagrams affect the maintainability of source code?: a family of experiments," *Empirical Software Engineering*, vol. 21, no. 1, pp. 212–259, 2016.

[35] T. DeMarco and T. Lister, *Programmer performance and the effects of the workplace*. IEEE, Aug. 1985.

[36] D. Fucci, G. Scanniello, S. Romano, M. Shepperd, B. Sigweni, F. Uyaguari, B. Turhan, N. Juristo, and M. Oivo, "An external replication on the effects of test-driven development using a multi-site blind analysis approach," in *Proc. of International Symposium on Empirical Software Engineering and Measurement*. ACM, 2016, pp. 3:1–3:10.

[37] J. Verelst, "The influence of the level of abstraction on the evolvability of conceptual models of information systems," in *Proc. of International Symposium on Empirical Software Engineering*. IEEE, 2004, pp. 17–26.