The Journal of Systems & Software 185 (2022) 111154

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Affective reactions and test-driven development: Results from three experiments and a survey☆

Maria Teresa Baldassarre [a,1], Danilo Caivano [a,1], Davide Fucci [b,1], Simone Romano [a,*,1], Giuseppe Scanniello [c,1]

[a] University of Bari, Bari, Italy
[b] Blekinge Institute of Technology, Karlskrona, Sweden
[c] University of Basilicata, Potenza, Italy

ARTICLE INFO

Article history:
Received 11 March 2021
Received in revised form 17 November 2021
Accepted 22 November 2021
Available online 7 December 2021

Keywords:
TDD
Affective state
SAM
Experiment
Replication
Survey

ABSTRACT

The research on the claimed effects of Test-Driven Development (TDD) on software quality and developers' productivity has shown inconclusive results. Some researchers have ascribed such results to the negative affective reactions that TDD would provoke when developers apply it. In this paper, we studied whether and in which phases TDD influences the affective states of developers, who are new to this development approach. To that end, we conducted a baseline experiment and two replications, and analyzed the data from these experiments both individually and jointly. Also, we performed *methodological triangulation* by means of an explanatory survey, whose respondents were experienced with TDD. The results of the baseline experiment suggested that developers like TDD significantly less, compared to a non-TDD approach. Also, developers who apply TDD like implementing production code significantly less than those who apply a non-TDD approach, while testing production code makes TDD developers significantly less happy. These results were not confirmed in the replicated experiments. We found that the moderator that better explains these differences across experiments is experience (in months) with unit testing, practiced in a test-last manner. The higher the experience with unit testing, the more negative the affective reactions caused by TDD. The results from the survey seem to confirm the role of this moderator.

1. Introduction

Test-Driven Development (TDD) is an incremental approach to software development where unit tests are written before production code (Beck, 2003). TDD promotes short cycles composed of three phases to implement functionality:

Red Phase. Write a unit test for a small chunk of functionality not yet implemented and watch the test fail;

Green Phase. Implement that chunk of functionality as quickly as possible and watch all unit tests pass;

Refactor Phase. Refactor the code and watch all unit tests pass.

TDD promises to increase external software quality (*i.e.,* less functional bugs) and developers' productivity because: *(i)* writing test first forces developers to break a problem into simpler ones; *(ii)* the tests provide initial software quality assurance; and *(iii)* the regression test suite resulting after several iterations allows the developer to catch breaking changes early. The safety net provided by the regression tests boosts developers' confidence to the extent that TDD is referred to as "the art of fearless programming" (Jeffries and Melnik, 2007).

The research on the claimed effects of TDD, gathered in secondary studies, has shown inconclusive results (*e.g.,* Turhan et al., 2010; Karac and Turhan, 2018). These inconclusive results can be due to the negative affective states that developers might experience when applying TDD. For example, the feeling of being unproductive due to the extra effort in writing unit tests rather than immediately focusing on the implementation of a functionality, or frustration due to the counter-intuitive *test-then-develop* order peculiar to TDD (*e.g.,* Erdogmus et al., 2010). Therefore, empirical evidence is needed to increase our body of knowledge on the affective states of developers due to TDD, as well as in which phases TDD influences their affective states. Such an increased body of knowledge could also explain why TDD is not

☆ Editor: Kelly Blincoe.

* Corresponding author.
E-mail addresses: mariateresa.baldassarre@uniba.it (M.T. Baldassarre), danilo.caivano@uniba.it (D. Caivano), davide.fucci@bth.se (D. Fucci), simone.romano@uniba.it (S. Romano), giuseppe.scanniello@unibas.it (G. Scanniello).

[1] The authors have equally contributed to the research presented in the paper.

https://doi.org/10.1016/j.jss.2021.111154
0164-1212/

widely practiced in open-source projects, as Beller et al. (2019) observed in their long-term study by monitoring for 2.5 years the development activities from 594 open-source projects.

In this paper, we present the results of three (controlled) experiments – *i.e.,* a baseline experiment (Romano et al., 2019), a first replication (Romano et al., 2020), and a second one – to study the affective reactions of developers when applying TDD. To complement and explain the results of the experiments, we perform an explanatory survey. In particular, we surveyed TDD experts (*i.e.,* TDD lecturers, developers, and/or researchers) to have a more critical perspective about some heterogeneous results across the experiments. In other words, our survey should be meant as a tool to perform *methodological triangulation* on those heterogeneous results across the three experiments.

In the baseline experiment (Romano et al., 2019), the participants were third-year undergraduates in Computer Science (CS). We asked them to carry out a development task by applying TDD or a non-TDD approach. After the task, we gathered the affective reactions of the participants by using *Self-Assessed Manikin* (*SAM*) (Bradley and Lang, 1994) complemented with a liking scale (Koelstra et al., 2012). SAM is a self-assessment instrument for measuring affective reactions to a stimulus in terms of pleasure, arousal, and dominance. We compared the affective reactions of the participants who used TDD with those of the participants who did not use TDD. We replicated the baseline experiment twice: in the first replication (Romano et al., 2020), the participants were second-year undergraduates in CS, while in the second replication, the participants were first-year graduates in CS. In each experiment, the participants that applied TDD were new to this development approach since they learned TDD in the course where that experiment was conducted. As for the survey, the respondents were attendees of the *2020 International Conference on Agile Software Development* (*XP2020*) having TDD experience.

In the baseline experiment (Romano et al., 2019), we observed that the participants liked TDD significantly less as compared to a non-TDD approach; the participants following TDD liked implementing production code significantly less than those following a non-TDD approach; and testing production code made the participants who followed TDD significantly less happy. The results from the first replication (Romano et al., 2020) failed to support the above-mentioned findings, as well as the results from the second replication and joint data analyses. We then conducted an exploratory analysis to identify moderators[2] that could explain the heterogeneous results across experiments. The experience (in months) with unit testing, practiced in a test-last manner, is the moderator that better explains the results. In particular, the experience with unit testing applied in a test-last manner: *(i)* causes more dislike towards TDD (as compared to a non-TDD approach); *(ii)* causes more dislike when implementing production code with TDD; and *(iii)* makes developers using TDD more unhappy when testing production code. In other words, the higher the experience with unit testing applied in a test-last manner, the more negative the affective reactions due to TDD. The results from the explanatory survey provided further support for findings *(i)* and *(iii)*.

This paper extends the ones by Romano et al. (*i.e.,* the baseline experiment, Romano et al., 2019 and the first replication, Romano et al., 2020) by adding the following new contributions:

- We conducted a new replication with first-year graduates in CS (who should be more experienced than undergraduates in CS).
- We performed joint data analyses to provide overall conclusions from the three experiments.
- We performed exploratory analyses to identify moderators that could explain potential heterogeneous results across the experiments.
- We conducted an explanatory survey to complement and explain the results of the three experiments.

[2] A moderator is a variable that is thought to modulate/temper the effect of an independent variable on a dependent one (Judd, 2001).

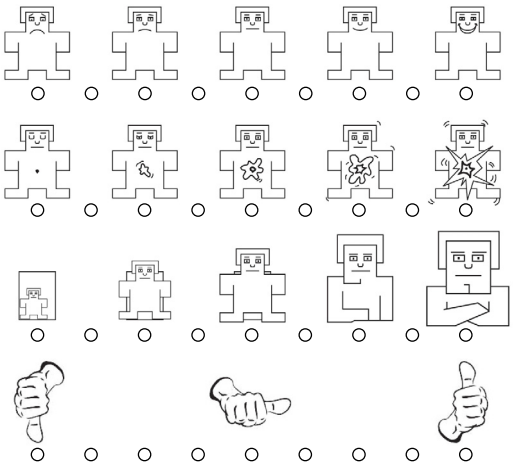


Fig. 1. From top down, the graphical representations of the pleasure, arousal, dominance, and liking dimensions.
Source: This figure has been taken from the paper by Romano et al. (2019).

Paper structure. In Section 2, we outline background information and work related to ours. We present our experiments in Section 3, while the results are outlined in Section 4. We discuss the obtained results, as well as possible threats to their validity, in Section 5. The survey and its results are presented in Sections 6 and 7, respectively. We discuss the results of the survey and their limitations in Section 8. Final remarks conclude the paper.

2. Background and related work

According to Russell and Mehrabian (1977), people's affective states can be characterized through three dimensions: pleasure, arousal, and dominance—*i.e.,* the *Pleasure–Arousal–Dominance* (*PAD*) model. Pleasure varies from unpleasant (*e.g.,* unhappy/sad) to pleasant (*e.g.,* happy/joyful), arousal ranges from inactive (*e.g.,* bored/calm) to active (*e.g.,* excited/stimulated), and dominance varies from "without control" to "in control of everything" (Koelstra et al., 2012). To measure a person's affective reaction to a stimulus in terms of PAD, Bradley and Lang (1994) proposed a pictorial self-assessment instrument they named SAM. This instrument graphically represents each PAD dimension and assigns a rating scale to it (see the first three rows in Fig. 1). For instance, SAM pictures the pleasure dimension through manikins varying from an unhappy manikin to a happy one. Thus, the nine-point rating scale placed just below the graphical representation of the pleasure dimension allows a person to self-assess, from one to nine, the pleasure dimension of their affective reaction. Recently, Koelstra et al. (2012) have complemented SAM with the liking dimension ranging from dislike – pictured through a thumb down – to like—pictured with a thumb up (see the last row in Fig. 1).

Both Human–Computer Interaction (HCI) and affective computing have used SAM in their empirical studies (*e.g.,* Koelstra et al., 2012; Herbon et al., 2005). Later, Software Engineering (SE) has used SAM. For example, Graziotin et al. (2013) conducted a study with eight developers who performed development tasks

on individual projects. Every ten minutes, the participants self-assessed both their affective state, by using SAM, and their productivity. The results show that pleasure and dominance are positively correlated with productivity.

Most SE studies have investigated the affective states of software users. For example, Curumsing et al. (2019) introduced an emotion-oriented requirements engineering approach where emotional states are modeled as users' goals. The authors validated their approach with several field studies in the context of smart homes for the elderly by following the participants' emotional reactions throughout the deployment of a smart-home platform. Stade et al. (2019) enhanced traditional data analytics with users' emotions to improve the insights regarding how users experience an app. In particular, they proposed a large-scale methodology to unobtrusively collect emotions from a large number of users. They reported a series of interviews with users showing encouraging results regarding emotion tracking during app usage, albeit with some minor privacy concerns.

When considering artifacts created and used by developers, Islam and Zibran (2018) and Mäntylä et al. (2016) explicitly considered the pleasure and arousal dimensions. Developers' emotions have been recognized using biofeedback devices connected to developers' bodies. For example, Müller and Fritz (2015) studied how developers' emotions and their progress implementing a task are related using biometric measures. In a laboratory study, the authors asked developers wearing three biometric sensors to self-assess their emotions. They then developed a machine learning classifier able to distinguish between positive and negative emotions with an accuracy of 71% and progress in the task with an accuracy of 67%. In an experiment with 28 academic practitioners, Fucci et al. (2019) showed that non-invasive biometrics can be used to predict the type of comprehension task a developer is engaged (*i.e.,* code vs. text comprehension) with an accuracy of 79%. Similar unobtrusive biometrics have been shown to measure the engagement of possible software users during requirements elicitation interviews (Girardi et al., 2020).

Although TDD has been the subject of several primary (*e.g.,* Beller et al., 2019; Fucci et al., 2016a, 2018) and secondary studies (*e.g.,* Turhan et al., 2010; Karac and Turhan, 2018), only Romano et al.'s studies (*i.e.,* the baseline experiment, Romano et al., 2019 and its first replication, Romano et al., 2020) focused on the affective reactions of developers when practicing TDD. We observed heterogeneous results across the baseline experiment and the first replication. We also speculated that the experience of developers with unit testing, applied in a test-last manner, can influence their affective reactions due to TDD. In particular, developers who have more months of experience with unit testing (applied in a test-last manner) can have affective reactions, due to TDD, which are more negative as compared to developers who have no/few months of experience.

3. Experiments

In this section, we describe the planning and execution of the baseline experiment and its two replications. We followed Wohlin et al.'s guidelines (Wohlin et al., 2012) to plan and execute these experiments.

3.1. Goals

We conducted our experiments to answer the following main Research Question (RQ):

RQ. Is there a difference in the affective reactions to using TDD or a non-TDD approach?

We formulated RQ to understand the affective reactions that TDD raises in developers in terms of pleasure, arousal, dominance, and liking. We also studied in which phases TDD would influence the affective reactions of developers. To this end, we formulated the following RQs:

RQ1. Is there a difference in the affective reactions of developers to implementing (production) code when applying TDD or a non-TDD approach?

RQ2. Is there a difference in the affective reactions of developers to testing (production) code when applying TDD or a non-TDD approach?

RQ3. Is there a difference in the affective reactions of developers to refactoring code when applying TDD and a non-TDD approach?

In particular, we aimed to understand the affective reactions of developers – in terms of pleasure, arousal, dominance, and liking – when they write production code (RQ1), test production code (RQ2), and refactor code (RQ3).

3.2. Participants

The participants in the baseline experiment (named BASILICATA) were 29 third-year undergraduates in CS at the University of Basilicata (Italy). The participants were taking the SE course when they took part in the experiment. All participants had passed the exams of Procedural Programming and Object-oriented Programming—*i.e.,* the courses about C and Java programming, in which refactoring principles were also provided being this practice part of day-to-day programming.

The participants in the first replication (named BARI) were 59 second-year undergraduates in CS at the University of Bari (Italy). The students were enrolled in the SE course when they participated in the replicated experiment. Before attending the course, the participants had passed the exams of the Programming I and Programming II courses, which focused on C and Java programming, along with refactoring principles.

The participants in the second replication (named SALERNO) were 13 first-year graduates in CS at the University of Salerno (Italy). The students were taking the Software Testing course when they were involved in the replication. The students had passed the following exams: Programming, Advanced Programming, Object-oriented Programming, and SE. These courses covered both basic and advanced notions of C and Java programming, along with software design and refactoring principles.

The students voluntarily participated in all the three experiments—*i.e.,* we did not pay them for their participation. As suggested by Carver et al. (2003), the students were rewarded for their participation, regardless of their performance, with two bonus points on the final mark of the course in which the experiment was executed. The participants were aware that their data would be confidentially treated and anonymously shared with the research community through an experimental package available on the web.[3]

The participants of each experiment had no knowledge of TDD before starting the course in which the experiment took place—in Section 3.7, we describe how the participants were trained to let them take part in the experiments.

[3] https://doi.org/10.6084/m9.figshare.15015837.v1.

3.3. Experimental material

In each experiment, we used two experimental objects: **Bowling Score Keeper** (**BSK**) and **Mars Rover** (**MR**). The participants who received BSK, in an experimental session, had to develop an API for calculating the score of a bowling game. The API allowed adding frames and bonus throws to a bowling game, identifying spare and strike frames, and computing the score of both frames and bowling game. On the other hand, the participants who worked on MR had to develop an API to move a rover on a planet. The planet was represented as a grid of cells. A cell could contain an obstacle that the rover could not go beyond. The API allowed moving the rover by using a string of basic commands (*i.e.,* turning left/right and moving forward/backward).

To develop BSK and MR, the participants had to write Java code. As for the unit tests, the participants had to use JUnit. Both BSK and MR are commonly used as experimental objects in empirical studies on TDD (*e.g.,* Fucci et al., 2016a, 2018, 2016b) and could be fulfilled in about three hours each (Fucci et al., 2018) (so mitigating a threat of *maturation* due to tiredness/boauredom Wohlin et al., 2012).

At the beginning of each experimental session, the students received the following material: *(i)* a brief description of the API to be implemented; *(ii)* a list of user stories concerning the features to be implemented; *(iii)* a template project for Eclipse containing stubs of the expected API signatures and a sample JUnit test class; and *(iv)* an acceptance test suite to simulate customers' acceptance of user stories.[4]

To gather the affective reactions of the participants, we used SAM (Bradley and Lang, 1994) and the liking scale (Koelstra et al., 2012). They allowed measuring affective reactions to a stimulus over nine-point rating scales.

3.4. Independent and dependent variables

In each experiment, we had two groups: *treatment* and *control*. The treatment group was made up of participants who had to use TDD to perform development tasks, while the control group was made up of participants who had to perform development tasks by using a non-TDD approach named YW (Your Way development)—*i.e.,* the approach each participant would normally utilize to develop (Fucci et al., 2018). Therefore, the main independent variable, or main factor, we manipulated in each experiment was **Approach**. This variable is nominal and assumes the following two values: *TDD* or *YW*.

To quantify PLeaSure (PLS), ARouSal (ARS), DOMinance (DOM), and LIKing (LIK) raised by the development APProach (APP), we used four dependent variables: $\texttt{APP}_{\texttt{PLS}}$, $\texttt{APP}_{\texttt{ARS}}$, $\texttt{APP}_{\texttt{DOM}}$, and $\texttt{APP}_{\texttt{LIK}}$. These variables assume integer values between one and nine because of the nine-point rating scales to measure affective reactions. To quantify the affective reactions (in terms of pleasure, arousal, dominance, and liking) to IMPlementing (IMP), TESting (TES), REFactoring (REF) code, we used the following dependent variables: $\texttt{IMP}_{\texttt{PLS}}$, $\texttt{IMP}_{\texttt{ARS}}$, $\texttt{IMP}_{\texttt{DOM}}$, $\texttt{IMP}_{\texttt{LIK}}$, $\texttt{TES}_{\texttt{PLS}}$, $\texttt{TES}_{\texttt{ARS}}$, $\texttt{TES}_{\texttt{DOM}}$, $\texttt{TES}_{\texttt{LIK}}$, $\texttt{REF}_{\texttt{PLS}}$, $\texttt{REF}_{\texttt{ARS}}$, $\texttt{REF}_{\texttt{DOM}}$, and $\texttt{REF}_{\texttt{LIK}}$.

3.5. Hypotheses

To answer our RQs, we formulated and tested the following parameterized null hypothesis:

***H0*$_X$.** There is no (statistically) significant difference between TDD and YW with respect to the dependent variable $X \in \{\texttt{APP}_{\texttt{PLS}}, \texttt{APP}_{\texttt{ARS}}, \texttt{APP}_{\texttt{DOM}}, \texttt{APP}_{\texttt{LIK}}, \texttt{IMP}_{\texttt{PLS}}, \texttt{IMP}_{\texttt{ARS}}, \texttt{IMP}_{\texttt{DOM}}, \texttt{IMP}_{\texttt{LIK}}, \texttt{TES}_{\texttt{PLS}}, \texttt{TES}_{\texttt{ARS}}, \texttt{TES}_{\texttt{DOM}}, \texttt{TES}_{\texttt{LIK}}, \texttt{REF}_{\texttt{PLS}}, \texttt{REF}_{\texttt{ARS}}, \texttt{REF}_{\texttt{DOM}}, \texttt{REF}_{\texttt{LIK}}\}$.

[4] We developed the acceptance test suite by using the Concordion framework.

3.6. Experimental design

BASILICATA and BARI. We used a *2 × 2 factorial* design (Wohlin et al., 2012). It is a kind of between-subjects design where each participant experimented only one development approach (*i.e.,* TDD or YW) on only one experimental object (*i.e.,* BSK or MR). This implies that we had four groups of participants—*i.e.,* the number of all possible combinations of development approaches and experimental objects. The assignment of the participants to the four groups was done randomly in both experiments. In BASILICATA, 15 participants were assigned to TDD– eight with BSK and seven with MR – while 14 participants were assigned to YW—seven with BSK and seven with MR. As for BARI, 28 participants were assigned to TDD– 14 with BSK and 14 with MR – while 31 participants were assigned to YW—16 with BSK and 15 with MR.

SALERNO. We used an *ABBA crossover* design (Vegas et al., 2016). It is a kind of a within-subjects design where each participant experimented each development approach only once. There are two *sequences* (*i.e.,* TDD-YW and YW-TDD), defined as the order with which the development approaches were administered to the participants, and two *periods* (*i.e.,* P1 and P2), defined as the times at which each approach was administered. The assignment of the participants to the sequences was done randomly. The participants in the sequence TDD-YW (*i.e.,* Group1) were six: they applied TDD (on BSK) in P1 and then YW (on MR) in P2. The participants in the sequence YW-TDD (*i.e.,* Group2) were seven: they applied YW (on BSK) in P1 and then TDD (on MR) in P2.

3.7. Experimental procedure

BASILICATA. The used experimental procedure follows.

1. We used a pre-questionnaire to collect students' availability to participate in BASILICATA, along with some demographic information (*e.g.,* months of experience with unit testing) about the participants.
2. All participants attended lessons on unit testing, JUnit, Test-Last Development (TLD), and Iterative-Test Last Development (ITLD). The participants also practiced unit testing with JUnit in a (laboratory) training session.
3. We randomly split the participants into the TDD and YW groups.
   - The participants in the TDD group attended a lesson on TDD and then applied this approach during two (laboratory) training sessions—*i.e.,* in each session, the participants developed a program by using TDD. The participants in the TDD group also carried out three homework assignments (*i.e.,* development tasks) by using TDD.
   - The participants in the YW group neither attended a lesson on TDD nor used this approach. These participants practiced TLD and ITLD in two (laboratory) training sessions and fulfilled the same homework as the TDD group but by practicing TLD and ITLD.

   It is worth mentioning that the participants in the TDD and YW groups underwent a similar training regardless of the group. This was to let them practice the administered development approaches as similarly as possible.
4. All participants took part in the experimental session in a laboratory—the laboratory was the same as the training sessions. All computers, in this laboratory, had the same hardware/software configuration. During the experimental session, the participants performed the development task by using the assigned development approach (*i.e.,*

TDD or YW) and then self-assessed, through SAM complemented with the liking scale, their affective reactions to: *(i)* the used development approach; *(ii)* implementing (production) code, *(iii)* testing (production) code; and *(iv)* refactoring code.

BARI. The experimental procedure used in this replication differs from that of BASILICATA in the third step. In particular, the participants in BARI fulfilled two homework assignments rather than three. We were forced to introduce this difference because the participants in BARI could not carry out a third homework assignment before the experiment took place. This was due to the strict teaching schedules and learning plans of the participants in BARI.

SALERNO. The experimental procedure was the same as BASILICATA except for the third and fourth steps, whose description follows:

3. All participants practiced TLD and ITLD in a (laboratory) training session and fulfilled a homework assignment by using TLD and ITLD. The participants then followed a lesson on TDD, practiced TDD in a training session, and fulfilled one homework assignment by using that development approach. We were forced to execute one training session with the participants and assign them one homework assignment per development approach due to the strict teaching schedules and learning plans of the participants.
4. As mentioned in Section 3.6, the participants were split into: Group1 and Group2. The participants in Group1 practiced TDD and then YW in two experimental sessions, while Group2 practiced YW and then TDD in those experimental sessions. The experimental sessions took place on different days in the same laboratory as the training session. The execution of the experimental sessions on two different days was to have an adequate washout period between the two laboratory sessions.

In any experiment, the time span between the execution of the first step (*i.e.,* filling in the pre-questionnaire) and the execution of the last step (*i.e.,* participating in the experimental session/s) was of about two weeks. It is also worth mentioning that, whatever the experiment was, the participants who applied YW, similarly to those who applied TDD, were asked to refactor their code every time they believed it was necessary. If a participant did not apply any refactoring, she would be asked not to rate her affective reaction to refactoring code (since she was not exposed to refactorings).

3.8. Analysis procedure

We analyzed the data from our experiments by following the procedure recommended by Santos et al. (2019) for analyzing groups of SE experiments. This procedure consists of four steps, each comprising two or more activities. A description of these steps and how we instantiated them in our analysis follows.

1. **Describe participants.** The objective of this step is to inform about the population which the results should be generalized to, as well as suggest possible sources of heterogeneity that may emerge when providing joint conclusions. This step consists of two activities.

   Activity 1.1—Provide summary statistics. For each experiment, we provided some descriptive statistics of the metrics (*e.g.,* months of experience in Java programming) used to assess the characteristics of the participants (*e.g.,* experience in Java programming).

   Activity 1.2—Provide profile plot. We used profile plots to show the characteristics of the participants averaged across the experiments.

2. **Analyze individual experiments.** The objective of this step is to provide findings from each individual experiment. This step is composed of the following activities.

   Activity 2.1—Provide summary statistics and visualizations. For each treatment (*i.e.,* TDD or YW) and each experiment, we provided descriptive statistics and used boxplots to describe the distribution of the dependent variable values.

   Activity 2.2—Provide profile plot. We used profile plots showing, for each treatment, the dependent variable values averaged across the experiments.

   Activity 2.3—Perform consistent individual analyses. For each experiment, we tested the defined null hypotheses introduced in Section 3.5. As suggested by Santos et al. (2019), we used a consistent method of statistical inference across the experiments. We applied the *ANOVA-Type Statistic* (*ATS*) method (Brunner et al., 2017), a non-parametric alternative to the *ANOVA* method. The ATS method was applied to test the null hypotheses in our previously published experiments (*i.e.,* BASILICATA, Romano et al., 2019 and BARI, Romano et al., 2020). This method has been recommended in HCI to analyze rating-scale data in factorial designs (Kaptein et al., 2010) (like the cases of BASILICATA and BARI). More recently, the ATS method has also been recommended in SE (where it is known as the *rank-based ANOVA-like* method) to analyze rating-scale data (Kitchenham et al., 2017). Due to the 2×2 factorial design of BASILICATA and BARI, the model underlying the ATS method had to take into account the Approach and *(Experimental) Object* factors as well as their interaction (Wohlin et al., 2012). As for SALERNO, we followed the procedure that Wellek and Blettner (2012) proposed for the analyses of ABBA crossover experiments. This procedure consists of two steps: *(i)* run a pre-test to check the assumption of negligible *carryover*[5] effect by using the *participant-wise sums* as the data; and *(ii)* run a test to check for significant differences between the treatments (*i.e.,* TDD vs. YW, in our case) by using the *participant-wise differences* as the data. Wellek and Blettner's procedure (Wellek and Blettner, 2012) allowed us to apply the ATS method to test the defined null hypotheses (as well as to check the assumption of negligible carryover effect), and thus perform a consistent inferential analysis across the experiments as Santos et al. (2019) suggested. As usual, we fixed the significance level, α, at 0.05—*i.e.,* if a *p*-value is less than 0.05, then there is a (statistically) significant difference.

3. **Aggregate results.** The goal of this step is to provide joint conclusions across the experiments. This step includes two activities.

   Activity 3.1—Apply AD. We used *Aggregated Data* (*AD*), which is also known as meta-analysis of effect sizes in SE. There are two kinds of meta-analysis models

[5] It is an internal validity threat, which occurs when a treatment is administered before the effect of a previously administered treatment has completely receded (Vegas et al., 2016).

– *fixed-* and *random-effects* (Borenstein et al., 2009) – and different strategies concerning how to apply these models based on the between-study heterogeneity (*e.g.,* Scanniello et al., 2018). Santos et al. (2019) recommended the use of random-effects meta-analysis models independently from the between-study heterogeneity. We followed such a recommendation when pooling the effect sizes from our experiments for the meta-analyses. Since rating-scale data (like ours) are often treated as continuous data in meta-analyses (Higgins et al., 2019), we used the *Standardized Mean Difference* (*SMD*) – also known as *Hedges'* (*adjusted*) *g* – as the effect size measure. It is worth noting that we computed the SMD values from
SALERNO as suggested by Madeyski and Kitchenham (2018) so making the SMD values from a crossover experiment comparable with those from between-participants experiments like BASILICATA and BARI.

Activity 3.2—Apply IPD-S. We exploited *Stratified Individual Participant Data* (*IPD-S*), which consists in jointly analyzing the data from all experiments by acknowledging which experiment the data come from. There are two kinds of IPD-S models: *fixed-* and *random-effects* (Whitehead, 2003). Wellek and Blettner (2012) recommended the use of random-effects IPD-S models. Commonly used random-effects IPD-S models are Linear Mixed Models (LMMs) with two factors: *Experiment* and *Treatment* (Whitehead, 2003). Accordingly, we used LMMs and modeled Experiment and Approach as the two factors. LMMs allowed us to analyze data from different kinds of experiments (*i.e.,* between- and within-participants experiments).

4. **Conduct exploratory analyses.** In case of heterogeneous results across the experiments, this step allows investigating the underlying reasons by identifying *experiment-* and *participant-level* moderators. This step includes the following activities.

Activity 4.1—Identify experiment-level moderators. We used AD (*i.e.,* we performed subgroup meta-analyses by using random-effects models) and IPD-S (*i.e.,* we fitted LMMs with interaction terms) in tandem to identify experiment-level moderators. When using LMMs, we fixed α at 0.1 as Santos et al. (2019) suggested.

Activity 4.2—Identify participant-level moderators. We used IPD-S (*i.e.,* we fitted LMMs with interaction terms) to identify participant-level moderators. Again, we fixed α at 0.1 (as suggested).

Activity 4.3—Acknowledge exploratory analysis limitations. We acknowledged the limitations of our exploratory analyses when discussing the limitations of our experiments in Section 5.2.

4. Experiment results

In this section, we present the results from our experiments based on the steps of the analysis procedure outlined in Section 3.8. We conclude the section by reporting the results of a quantitative further analysis.

Table 1
Mean (and SD) of the months of experience (in Java programming, as professional developer, and in unit testing) of the participants in each experiment.

Experiment	Months of experience		
	Java programming	Professional developer	Unit testing
BASILICATA	9.310 (4.115)	0 (0)	4.724 (4.651)
BARI	18.893 (28.060)	2.036 (8.038)	1.714 (2.484)
SALERNO	21.538 (10.556)	4 (6.671)	0 (0)

4.1. Description of participants

In Table 1, we show mean and Standard Deviation (SD) of the months of experience in Java programming, as professional developer, and in unit testing that the participants had in each experiment (Activity 1.1). The profile plot in Fig. 2 graphically depicts the months of experience averaged across the experiments (Activity 1.2). We can observe that the participants with the highest experience in Java programming and as professional developers are those in SALERNO (on average, 21.538 and 4 months, respectively) followed by the participants in BARI (on average, 18.893 and 2.036 months, respectively). As for the experience in unit testing, the most experienced participants are those in BASILICATA (on average, 4.724), followed by BARI (on average, 1.714). It is worth mentioning that the participants were not knowledgeable on TDD when we gathered their experience in unit testing—*i.e.,* about two weeks before the experimental session/s took place (see Section 3.7). Therefore, they were used to practice unit testing in a test-last manner. That is, they were used to write unit tests after they had written production code. We can also notice that, although the participants in SALERNO were graduates, they had no experience with unit testing—as opposed to the participants in BASILICATA and BARI that, although undergraduates, had more experience with unit testing. This difference can be explained by the different university curricula: in SALERNO, unit testing was one of the topics of the Unit Testing course—*i.e.,* the course in which the experiment was conducted. When the participants in SALERNO filled in the pre-questionnaire, they had no experience with unit testing.

Summing up, the results shown in Table 1 and Fig. 2 suggest the presence of heterogeneity in the averaged experience of the participants across the experiments, which could cause between-study heterogeneity when providing joint conclusions.

4.2. Analyses of individual experiments

In Table 2, we report mean and SD summarizing the dependent variable values for each development approach across the experiments. In Fig. 3, we show the boxplots summarizing the values of the dependent variables grouped by experiment and treatment (Activity 2.1). The profile plots in Fig. 4 show the mean values of the dependent variables of each approach across the experiments (Activity 2.2). Finally, in Table 3, we report the *p*-values for the Approach factor that the ATS method returned for each dependent variable and experiment (Activity 2.3). In the rest of this subsection, we present the results of the experiments individually.

4.2.1. BASILICATA

Affective reactions to development approach. The boxplots in Fig. 3.a do not show a huge difference in the affective reactions to TDD and YW in terms of pleasure ($\texttt{APP}_{\texttt{PLS}}$), arousal ($\texttt{APP}_{\texttt{ARS}}$), and dominance ($\texttt{APP}_{\texttt{DOM}}$) – the boxes mostly overlap one another – despite we can notice that the participants who used YW were slightly more positive than those who used TDD with respect

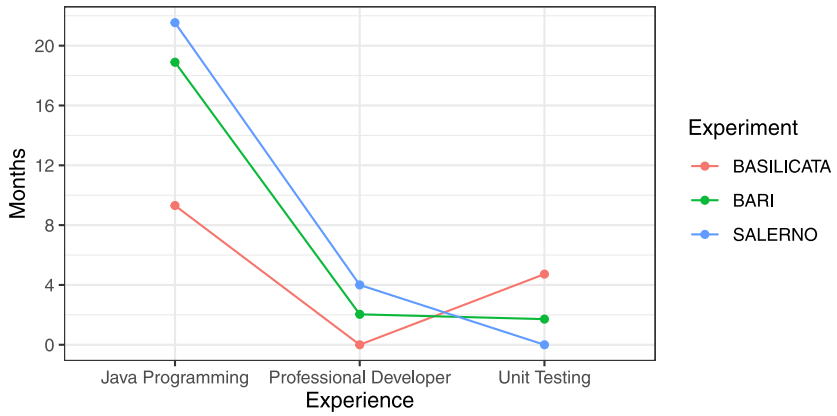


Fig. 2. Profile plot showing the months of experience (in Java programming, as professional developer, and in unit testing) of the participants averaged across the experiments.

Table 2
Mean (and SD) for each dependent variable grouped by approach and experiment.

Dep. Var.	Approach	Experiment		
		BASILICATA	BARI	SALERNO
APP_{PLS}	TDD	4.933 (1.87)	6.036 (2.236)	6.538 (2.222)
	YW	6.357 (1.692)	6.065 (1.389)	6.769 (1.641)
APP_{ARS}	TDD	5.333 (1.447)	5.321 (1.722)	6.154 (1.908)
	YW	6 (1.468)	5.097 (1.557)	6 (2.16)
APP_{DOM}	TDD	5.4 (1.549)	6.286 (2.307)	7.154 (2.193)
	YW	6.214 (1.762)	6.581 (1.689)	6.462 (1.613)
APP_{LIK}	TDD	5.2 (1.656)	6.214 (2.699)	7.308 (1.548)
	YW	7 (1.24)	6.613 (1.783)	6.308 (2.394)
IMP_{PLS}	TDD	5.2 (1.474)	6.143 (1.995)	7 (1.78)
	YW	6.143 (2.143)	6.097 (2.181)	7 (1.633)
IMP_{ARS}	TDD	5.533 (1.125)	5.786 (1.893)	6.231 (2.242)
	YW	5.929 (1.979)	5.903 (2.087)	6.385 (1.446)
IMP_{DOM}	TDD	4.933 (1.534)	6.536 (2.027)	7.615 (0.87)
	YW	5.857 (2.143)	6.484 (2.08)	6.462 (1.898)
IMP_{LIK}	TDD	5.267 (1.534)	6.607 (2.166)	7.615 (0.87)
	YW	6.714 (1.729)	6.742 (2.113)	6.692 (2.136)
TES_{PLS}	TDD	4.867 (1.685)	5.786 (2.5)	6.385 (2.022)
	YW	6.357 (1.393)	5.806 (1.642)	6 (2.041)
TES_{ARS}	TDD	5.4 (1.352)	5.214 (1.833)	6 (1.826)
	YW	6.071 (1.685)	5.258 (2.144)	6.077 (1.382)
TES_{DOM}	TDD	5.467 (1.506)	6.286 (2.225)	6.769 (1.964)
	YW	5.857 (1.46)	6.774 (1.765)	6.846 (2.115)
TES_{LIK}	TDD	5.533 (1.685)	5.75 (2.548)	6.538 (1.984)
	YW	6.714 (1.326)	6.516 (1.313)	5.923 (2.1)
REF_{PLS}	TDD	5.333 (1.118)	5.52 (2.044)	6.846 (1.772)
	YW	5.846 (1.573)	5.448 (1.429)	6.727 (1.902)
REF_{ARS}	TDD	5.667 (1.581)	5.24 (1.985)	6.846 (1.772)
	YW	5.538 (2.025)	5.138 (1.807)	6.545 (2.018)
REF_{DOM}	TDD	5.778 (1.481)	6.16 (2.055)	7.231 (1.092)
	YW	6 (1.732)	6.172 (1.311)	6.545 (1.635)
REF_{LIK}	TDD	5.333 (1.118)	5.92 (1.998)	7.154 (1.405)
	YW	6.308 (1.702)	5.793 (1.264)	6.636 (1.748)

to the PAD dimensions (*e.g.,* Fig. 4.a). The results from the ATS method in Table 3 do not indicate significant differences between TDD and YW—no p-value is less than 0.05. Therefore, we cannot reject the corresponding null hypotheses (*i.e.,* $H0_{APP_{PLS}}$, $H0_{APP_{ARS}}$, and $H0_{APP_{DOM}}$). As for the affective reactions to TDD and YW in terms of liking (APP_{LIK}), the boxplot in Fig. 3.a suggests that the participants in the YW group liked the experienced approach more than the participants in the TDD group (7 vs. 5.2 on average, as shown in Table 2). The p-value (0.002) returned by the ATS method (Fig. 3) allows us to reject $H0_{APP_{LIK}}$ and then accept the alternative hypothesis: there is a significant difference between TDD and YW (in favor of YW) with respect to APP_{LIK}.

Affective reactions to implementing code. Fig. 3.b does not show remarkable differences between TDD and YW for pleasure (IMP_{PLS}), arousal (IMP_{ARS}), and dominance (IMP_{DOM}) when considering the implementation of production code, although there is a slight trend in favor of YW (Fig. 4.b). The results in Table 3 seem to confirm that there is not a significant difference between TDD and YW regarding these dimensions. Accordingly, the corresponding null hypotheses cannot be rejected (*i.e.,* $H0_{IMP_{PLS}}$, $H0_{IMP_{ARS}}$, and $H0_{IMP_{DOM}}$). As far as liking (IMP_{LIK}) is concerned, Fig. 3.b suggests that the participants who applied YW liked implementing code more than those who applied TDD (the mean values are 6.714 and 5.267, see Table 2). The p-value (0.04) returned by the ATS method (Table 3) confirms that there is a significant difference between TDD and YW (in favor of YW) with respect to IMP_{LIK}—*i.e.,* we can reject $H0_{IMP_{LIK}}$ and then accept the alternative hypothesis.

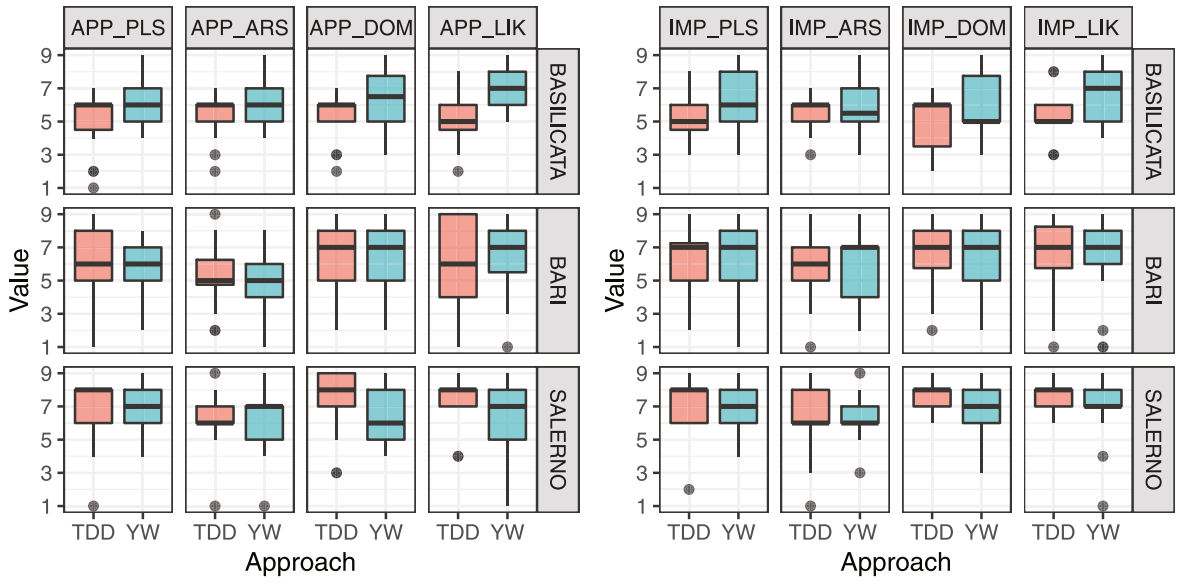
Affective reactions to testing code. When considering arousal (TES_{ARS}) and dominance (TES_{DOM}) to testing code, we cannot notice substantial differences between the two development approaches (Fig. 3.c). On the contrary, when considering liking (TES_{LIK}), we can notice a more evident difference between TDD and YW (Figs. 3.c and 4.c) in favor of YW. However, the results from the ATS method (Table 3) do not allow rejecting the respective null hypotheses (*i.e.,* $H0_{TES_{ARS}}$, $H0_{TES_{DOM}}$, and $H0_{TES_{LIK}}$). By looking at Fig. 3.c, we can notice that there is a remarkable difference between TDD and YW in terms of pleasure (TES_{PLS}). The participants using YW were happier than those using TDD when testing code (6.357 vs. 4.867 on average, see Table 2). The p-value (0.018) returned by the ATS method allows rejecting $H0_{TES_{PLS}}$ and then accepting the alternative hypothesis: there is a significant difference between TDD and YW (in favor of YW) with respect to TES_{PLS}.

Affective reactions to refactoring code. The boxplots in Fig. 3.d do not highlight substantial differences between TDD and YW in terms of pleasure (REF_{PLS}), arousal (REF_{ARS}), dominance (REF_{DOM}), and liking (REF_{LIK}) when refactoring code—whatever the dependent variable is, the box for TDD mostly overlaps with the one for YW. The ATS method does not allow rejecting the null hypotheses (Table 3).

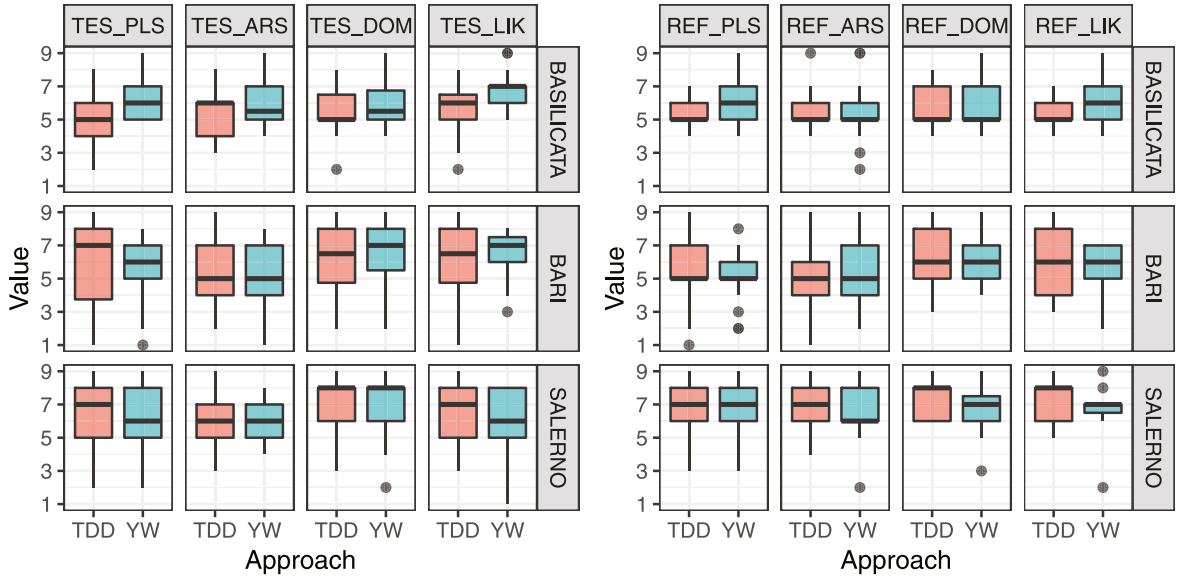
Further results. The p-values returned by the ATS method for Object and the Approach × Object interaction (which we did not report in Table 3 for the sake of readability) are always no less than 0.05.

4.2.2. BARI

Affective reactions to development approach. The boxplots in Fig. 3.a do no show huge differences in the affective reactions to the used development approach in terms of all studied dimensions. The p-values returned by the ATS method (Table 3) are all not less than 0.05, thus we cannot reject any null hypothesis.



(a) Affective reactions to development approach. (b) Affective reactions to implementation code.



(c) Affective reactions to testing code. (d) Affective reactions to refactoring code.

Fig. 3. Boxplots summarizing the dependent variable values for each approach in each experiment.

Affective reactions to implementing code. As shown in Fig. 3.b, there is no remarkable difference between TDD and YW regarding the studied dimensions of the affective reactions to implementing code. The results from the ATS method (Table 3) do not suggest any significant difference between TDD and YW—no p-value is less than 0.05.

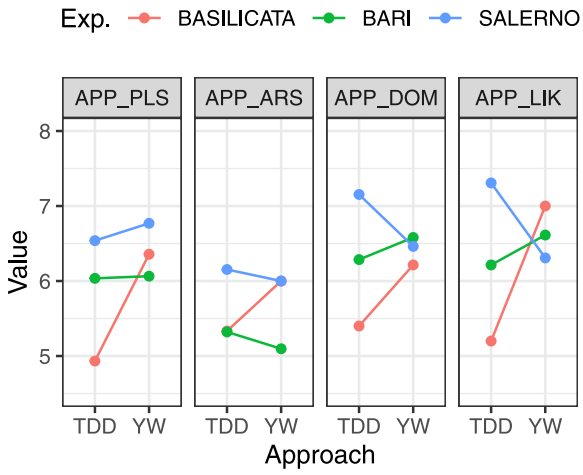
Affective reactions to testing code. The boxplots in Fig. 3.c show that, regardless of the dimension, the affective reactions of the YW group to testing code are quite similar to those of the TDD group. In no case, the results from the ATS method (Table 3) allow rejecting the null hypotheses.

Affective reactions to refactoring code. By looking at the boxplots in Fig. 3.d, no noticeable difference emerges between TDD and YW in terms of the dimensions of affective reactions to refactoring code. The p-values of the ATS method (see Table 3) do not indicate the presence of a significant difference between TDD and YW.

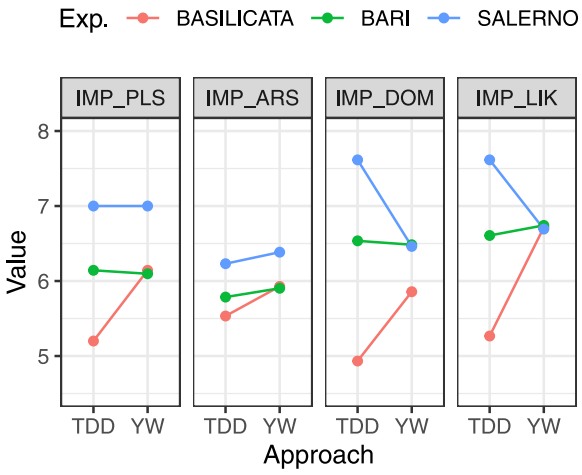
Further results. For $\texttt{APP}_{\texttt{LIK}}$, the p-value for Object is 0.032 (not reported in Table 3 for readability reasons), suggesting that the used experimental objects significantly influenced the affective reactions of the participants to the development approach in terms of liking. However, the effect of the experimental objects is consistent within both TDD and YW groups since the Approach $\times$ Object interaction is not significant.

4.2.3. SALERNO

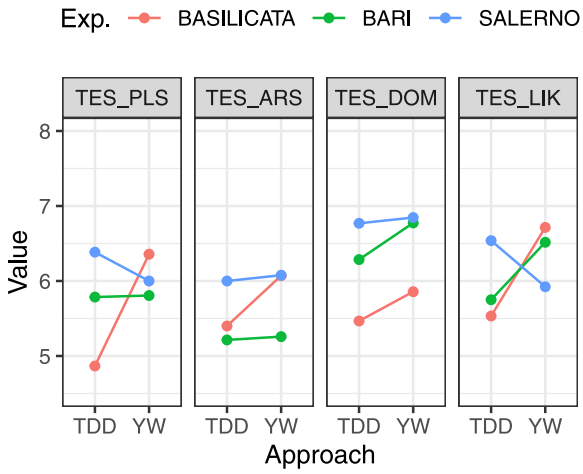
Affective reactions to development approach. The boxplots in Fig. 3.a suggest no substantial difference in the affective reactions to TDD and YW in terms of the considered dimensions. The p-values returned by the ATS method (Table 3) are not less than 0.05.



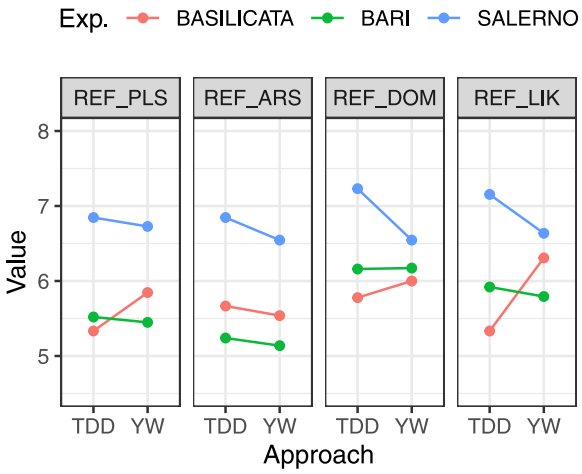
(a) Affective reactions to development approach.



(b) Affective reactions to implementing code.



(c) Affective reactions to testing code.



(d) Affective reactions to refactoring code.

Fig. 4. Profile plots showing the mean values for each dependent variable and each approach across the experiments.

Table 3
p-values for the Approach factor that the ATS method returned for each dependent variable in each experiment. *p*-values less than $\alpha = 0.05$ are highlighted in bold.

Dep. var.	BASILICATA		BARI		SALERNO	
	p-value	$H0_X$ outcome	*p*-value	$H0_X$ outcome	*p*-value	$H0_X$ outcome
APP_{PLS}	0.162	Failed to reject	0.694	Failed to reject	0.783	Failed to reject
APP_{ARS}	0.277	Failed to reject	0.642	Failed to reject	0.78	Failed to reject
APP_{DOM}	0.28	Failed to reject	0.83	Failed to reject	0.26	Failed to reject
APP_{LIK}	**0.002**	Rejected in favor of YW	0.921	Failed to reject	0.095	Failed to reject
IMP_{PLS}	0.201	Failed to reject	0.904	Failed to reject	0.779	Failed to reject
IMP_{ARS}	0.68	Failed to reject	0.778	Failed to reject	0.685	Failed to reject
IMP_{DOM}	0.345	Failed to reject	0.953	Failed to reject	**0.005**	Rejected in favor of TDD
IMP_{LIK}	**0.04**	Rejected in favor of YW	0.805	Failed to reject	**0.01**	Rejected in favor of TDD
TES_{PLS}	**0.018**	Rejected in favor of YW	0.572	Failed to reject	0.364	Failed to reject
TES_{ARS}	0.415	Failed to reject	0.745	Failed to reject	0.833	Failed to reject
TES_{DOM}	0.634	Failed to reject	0.509	Failed to reject	0.945	Failed to reject
TES_{LIK}	0.05	Failed to reject	0.459	Failed to reject	0.26	Failed to reject
REF_{PLS}	0.548	Failed to reject	0.918	Failed to reject	0.527	Failed to reject
REF_{ARS}	0.892	Failed to reject	0.908	Failed to reject	0.606	Failed to reject
REF_{DOM}	0.847	Failed to reject	0.993	Failed to reject	0.239	Failed to reject
REF_{LIK}	0.225	Failed to reject	0.92	Failed to reject	0.08	Failed to reject

Affective reactions to implementing code. Fig. 3.b does not show remarkable differences between TDD and YW for pleasure and arousal when studying the implementation of the production code. The *p*-values returned by the ATS method (Table 3) seem

to confirm that there is not a significant difference between TDD and YW regarding these dimensions. With respect to dominance and liking, Fig. 3.b suggests that the participants who applied TDD were more in-control when implementing code, as compared to the ones who applied YW (on average 7.615 vs. 6.462, see Table 2), and also more liked implementing code (on average 7.615 vs. 6.692). The *p*-values (Table 3) for $\texttt{IMP}_{\texttt{DOM}}$ and $\texttt{IMP}_{\texttt{LIK}}$ are 0.005 and 0.01, respectively, so allowing us to reject $H0_{\texttt{IMP}_{\texttt{DOM}}}$ and $H0_{\texttt{IMP}_{\texttt{LIK}}}$. Therefore, there is a significant difference between TDD and YW (in favor of TDD) with respect to $\texttt{IMP}_{\texttt{DOM}}$ and $\texttt{IMP}_{\texttt{LIK}}$.

Affective reactions to testing code. The boxplots in Fig. 3.c show that the affective reactions of the YW group to testing code are quite similar to those of the TDD group, whatever the dimension is. The results reported in Table 3 do not allow rejecting the null hypotheses.

Affective reactions to refactoring code. As Fig. 3.d suggests, there is no noticeable difference between TDD and YW when refactoring code, regardless of the dimension considered. The *p*-values in Table 3 do not indicate the presence of significant differences between TDD and YW, and thus we cannot reject the null hypotheses.

Further results. We met the assumption of negligible carry-over effect for all dependent variables—*i.e.,* in no case, the pre-test returned a *p*-value less than 0.05 (not reported in Table 3 for readability reasons). (See Fig. 6.)

4.3. Aggregation of results

In Fig. 5, we show the results of our meta-analyses – a meta-analysis for each dependent variable – through forest plots (Activity 3.1). Each forest plot reports, besides the SMD estimates and the corresponding 95% Confidence Intervals (CIs), two statistics to assess the between-study heterogeneity: the *p*-value from the *Q-test* and I^2 statistic (Borenstein et al., 2009). The *Q*-test allows determining whether the between-study heterogeneity is (statistically) significant—the most used α value is 0.1 (Scanniello et al., 2018). Therefore, if the *Q*-test *p*-value is lower than 0.1, we assume that the between-study heterogeneity is (statistically) significant. We use the I^2 statistic to measure the extent of the between-study heterogeneity[6] when the *Q*-test reveals that the between-study heterogeneity is significant (Scanniello et al., 2018). As for the IPD-S results, they are summarized in Table 4 (Activity 3.2). In particular, for each dependent variable, Table 4 shows the LMM estimates for TDD, YW, and their Mean Difference (MD). The CI and *p*-value of each estimate are also shown. In the following of this subsection, we report the aggregated results by RQ.

4.3.1. RQ—Affective reactions to development approach

As Fig. 5.a shows, the affective reactions to the development approach in terms of pleasure ($\texttt{APP}_{\texttt{PLS}}$) are more positive for YW than for TDD in any experiment. The joint effect (SMD = −0.232), in favor of YW, is *small*[7] and not significant—the CI spans 0. The IPD-S results shown in Table 4 confirm that, overall, pleasure due to the development approach is greater for YW, but the joint effect (MD = −0.473) is not significant—the *p*-value (0.331) is not less than 0.05. Regarding arousal ($\texttt{APP}_{\texttt{ARS}}$) due to the development approach, the joint SMD (−0.011) is nearly null and not significant (Fig. 5.a). The IPD-S results (Table 4) confirm that the joint effect on $\texttt{APP}_{\texttt{ARS}}$ is nearly null (MD = 0.015) and not significant (*p*-value = 0.959). As for dominance ($\texttt{APP}_{\texttt{DOM}}$), the overall SMD (−0.118) is in favor of YW and indicates a negligible and not significant joint effect (Fig. 5.a). The IPD-S results confirms these outcomes (Table 4). As Fig. 5.a shows, the between-study heterogeneity is not significant for $\texttt{APP}_{\texttt{PLS}}$, $\texttt{APP}_{\texttt{ARS}}$, and $\texttt{APP}_{\texttt{DOM}}$—the *Q*-test p-values (0.25, 0.434, and 0.304, respectively) are all not less than 0.1. As for the affective reactions to the development approach in terms of liking ($\texttt{APP}_{\texttt{LIK}}$), we can notice that the joint effect (SMD = −0.279) is in favor of YW, it is small and not significant (see Fig. 5.a). The results in Table 4 confirm that the joint effect (MD = −0.38) is in favor of YW and not significant. Finally, Fig. 5.a suggests a significant and substantial/considerable between-study heterogeneity (*p*-value = 0.01 and $I^2 = 82\%$). Therefore, moderators should be identified to explain the detected between-study heterogeneity. As suggested by Santos et al. (2019), we analyzed these moderators in the fourth step of the analysis procedure and show the results of these analyses in Section 4.4.

> **Answer to RQ.** Although we found that the participants in BASILICATA liked YW significantly more than TDD, the overall conclusion is that there is no significant difference between the affective reactions of developers when using YW or TDD.

4.3.2. RQ1—Affective reactions to implementing code

As for $\texttt{IMP}_{\texttt{PLS}}$ and $\texttt{IMP}_{\texttt{ARS}}$, the joint effects, both in favor of YW, are negligible and not significant as shown in Fig. 5.b and Table 4. The between-study heterogeneity for both $\texttt{IMP}_{\texttt{PLS}}$ and $\texttt{IMP}_{\texttt{ARS}}$ is not significant as the *Q*-test *p*-values in Fig. 5.b suggest. As for $\texttt{IMP}_{\texttt{DOM}}$ and $\texttt{IMP}_{\texttt{LIK}}$, the overall SMD values (0.101 and −0.101, respectively) indicate negligible effects—the former in favor of TDD, the latter in favor of YW. These effects are both not significant. The IPD-S results in Table 4 are consistent with those above-mentioned. Finally, the Q-test p-values reported in Fig. 5.b indicate a significant between-study heterogeneity for both $\texttt{IMP}_{\texttt{DOM}}$ and $\texttt{IMP}_{\texttt{LIK}}$. Based on the I^2 values, the between-study heterogeneity is substantial for $\texttt{IMP}_{\texttt{DOM}}$, while substantial/considerable for $\texttt{IMP}_{\texttt{LIK}}$. Therefore, we need to identify moderators for both $\texttt{IMP}_{\texttt{DOM}}$ and $\texttt{IMP}_{\texttt{LIK}}$ (see Section 4.4).

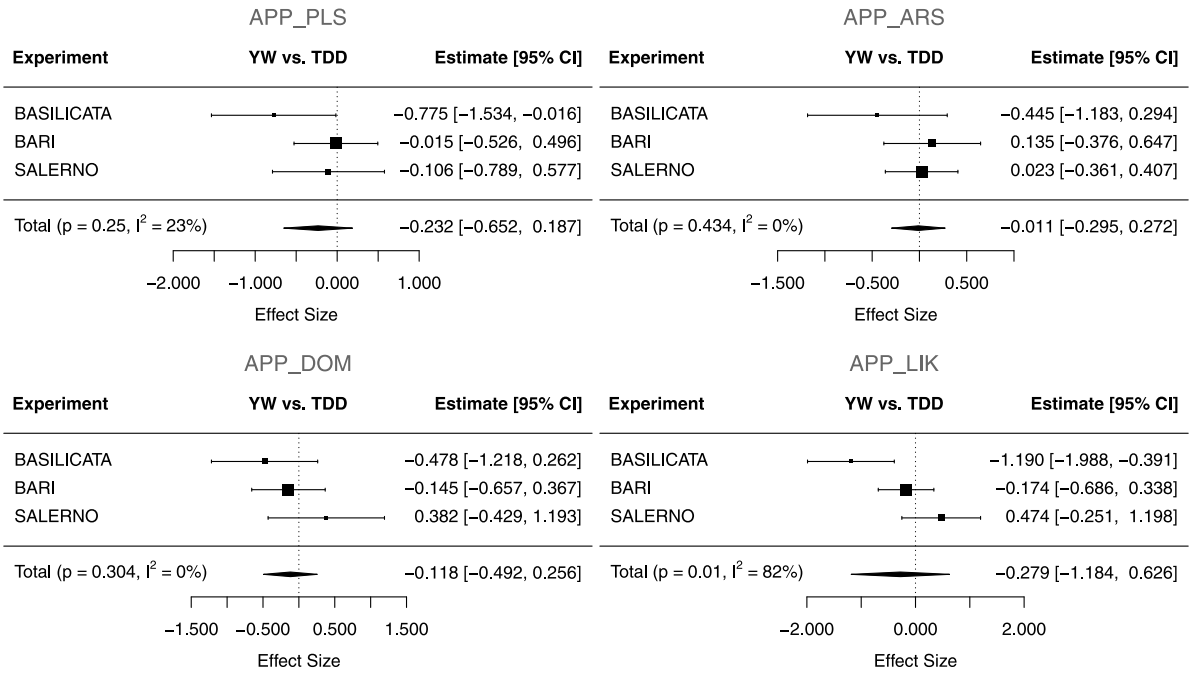
> **Answer to RQ1.** We found that the participants in BASILICATA who used TDD liked implementing production code significantly less than those who used YW, while the participants in SALERNO who applied TDD were significantly more in-control when implementing production code and also liked implementing production code significantly more. However, the joint conclusion does not indicate a significant difference between the development approaches in terms of the affective reactions of developers to implementing production code.

4.3.3. RQ2—Affective reactions to testing code

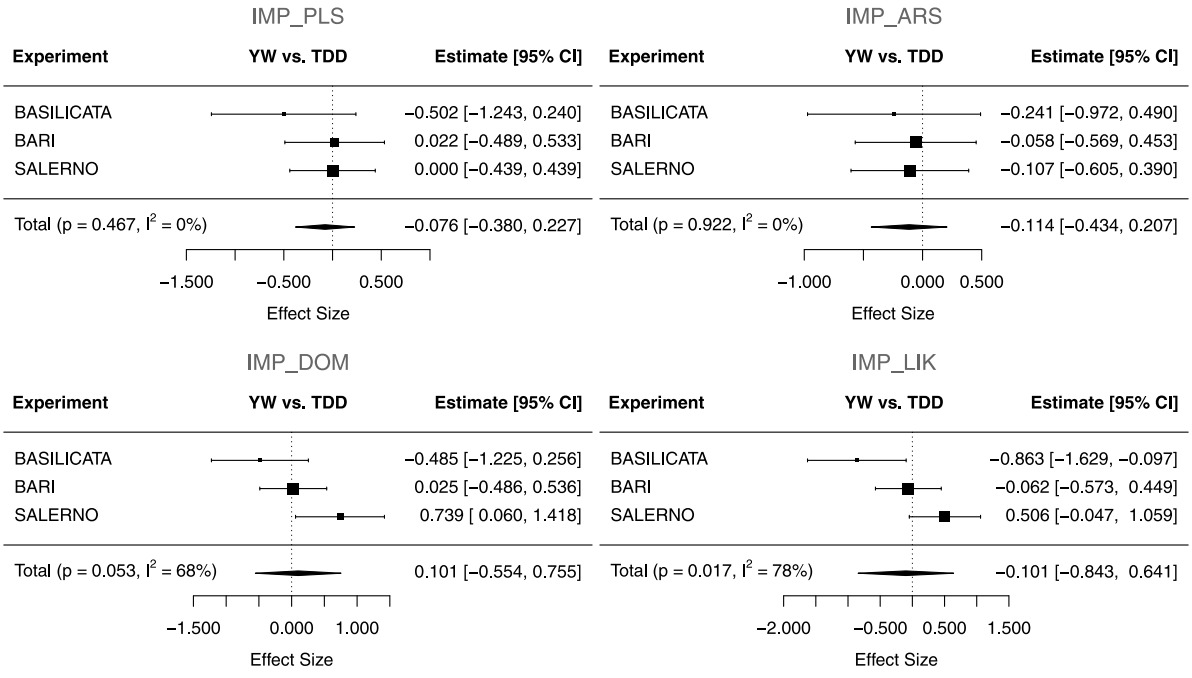
As shown in Fig. 5.c, the joint effects of the development approach on $\texttt{TES}_{\texttt{PLS}}$ (SMD = −0.174), $\texttt{TES}_{\texttt{ARS}}$ (SMD = −0.118), and $\texttt{TES}_{\texttt{DOM}}$ (SMD = −0.174) are in favor of YW. The effects are negligible and not significant. The IPD-S results (see both MD p-values and estimates in Table 4) confirm that these effects are in favor of YW and not significant. By looking at Fig. 5.c, we can also observe a significant (*Q*-test *p*-value = 0.038) and substantial between-study heterogeneity ($I^2 = 74\%$) for $\texttt{TES}_{\texttt{PLS}}$. This outcome deserves an exploratory analysis to identify moderators (see Section 4.4). Regarding both $\texttt{TES}_{\texttt{ARS}}$ and $\texttt{TES}_{\texttt{DOM}}$, the *Q*-test *p*-values indicate that the between-study heterogeneity is not

[6] According to Higgins et al.'s guidelines (Higgins et al., 2019), the between-study heterogeneity can be interpreted as: *unimportant*, if $I^2 \leq 40\%$; *moderate*, if $30\% \leq I^2 \leq 60\%$; *substantial*, if $50\% \leq I^2 \leq 90\%$; or *considerable*, if $I^2 \geq 75\%$.

[7] According to Cohen's guidelines (Cohen, 1992), the SMD can be considered: *negligible*, if |SMD| < 0.2; small, if 0.2 ≤ |SMD| < 0.5; *medium*, if 0.5 ≤ |SMD| < 0.8; or *large*, otherwise.



(a) Affective reactions to development approach.



(b) Affective reactions to implementing code.

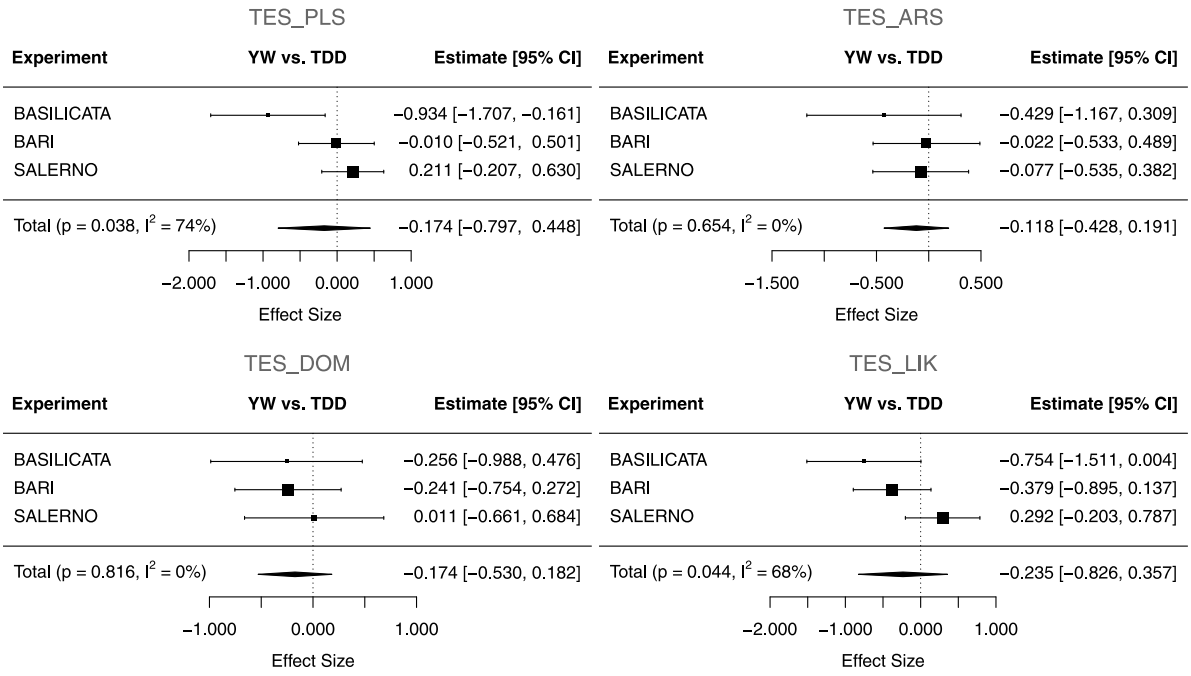
Fig. 5. Forrest plots depicting the SMD values for each dependent variable.

significant. Finally, the joint effect of the development approach on $\texttt{TES}_{\texttt{LIK}}$ (SMD = −0.279) is in favor of YW, small, and not significant as Fig. 5.c suggests. These results are consistent with the IPD-S results shown in Table 4. Moreover, there is a significant and substantial/considerable between-study heterogeneity for $\texttt{TES}_{\texttt{LIK}}$ (Fig. 5.c). Accordingly, we must identify moderators (see Section 4.4).

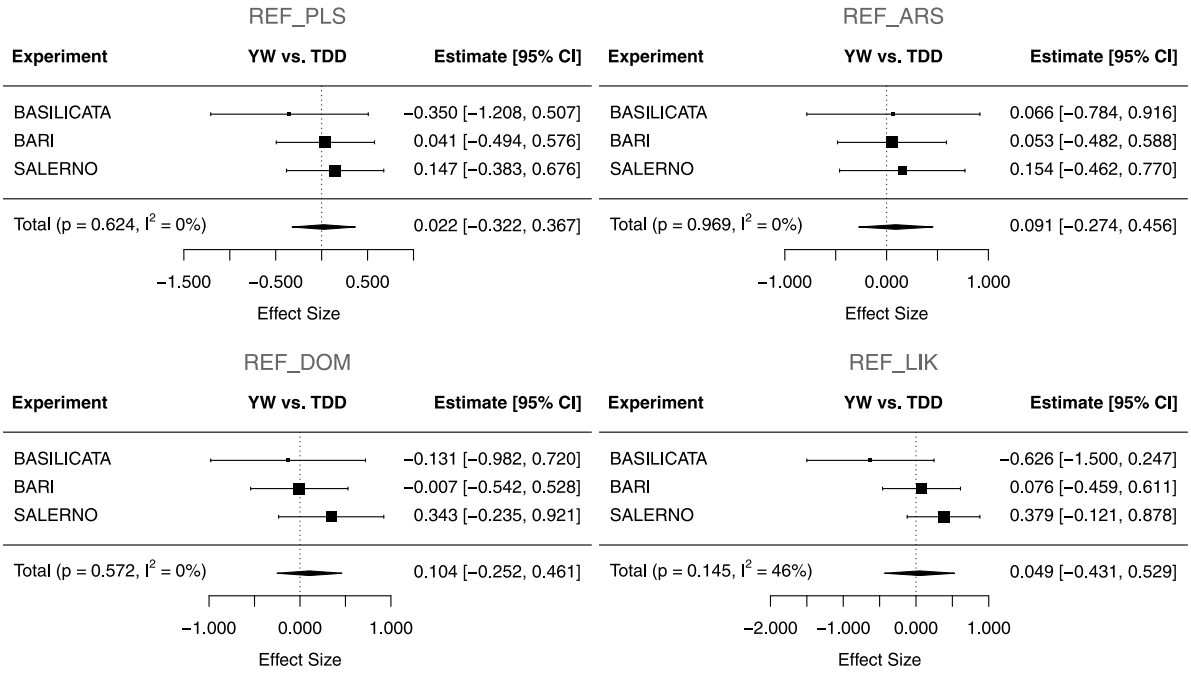
Answer to RQ2. In BASILICATA, we found that testing production code made the participants using TDD significantly less happy. However, the joint conclusion does not indicate a significant difference between the development approaches in terms of the affective reactions of developers to testing production code.

4.3.4. RQ3—Affective reactions to refactoring code

As Fig. 5.d shows, the joint effects of the development approach on the affective reactions to refactoring code are in favor of TDD for any dimension—*e.g.*, for $\texttt{REF}_{\texttt{PLS}}$, the overall SMD value



(c) Affective reactions to testing code.



(d) Affective reactions to refactoring code.

Fig. 5. (*continued*).

is 0.022. The effects are negligible and not significant for any dimension. The IPD-S results in Table 4 confirm that these effects are not significant. In no case, the Q-test p-values in Fig. 5.d suggest a significant between-study heterogeneity.

Answer to RQ3. The joint conclusion does not indicate a significant difference between the development approaches in terms of the affective reactions to refactoring code.

4.4. (Quantitative) exploratory analyses

Since the participants in both BASILICATA and BARI were undergraduates and those in SALERNO were graduates, a potential experiment-level moderator is *Graduate—i.e.,* a binary factor that assumes 0 if the student is undergraduate, while 1 otherwise. In Fig. 6, we show the results of our subgroup meta-analyses by using forest plots. It is worth recalling that, in these subgroup meta-analyses, we focused on the dependent variables for which we observed a significant between-study heterogeneity, namely: $\texttt{APP}_{\texttt{LIK}}$, $\texttt{IMP}_{\texttt{DOM}}$, $\texttt{IMP}_{\texttt{LIK}}$, $\texttt{TES}_{\texttt{PLS}}$, and $\texttt{TES}_{\texttt{LIK}}$. We also built LMMs

Table 4
IPD-S results for each dependent variable. p-values less than $\alpha = 0.05$ are highlighted in bold.

Dep. var.	TDD		YW		MD	
	Estimate [95% CI]	p-value	Estimate [95% CI]	p-value	Estimate [95% CI]	p-value
APP_{PLS}	5.81 [4.976, 6.644]	**0**	6.275 [5.793, 6.756]	**0**	−0.473 [−1.489, 0.544]	0.331
APP_{ARS}	5.569 [5.002, 6.136]	**0**	5.556 [4.979, 6.132]	**0**	0.015 [−0.613, 0.643]	0.959
APP_{DOM}	6.262 [5.32, 7.204]	**0**	6.465 [5.96, 6.97]	**0**	−0.202 [−1.297, 0.893]	0.695
APP_{LIK}	6.235 [5.061, 7.409]	**0**	6.617 [5.991, 7.242]	**0**	−0.38 [−2.099, 1.338]	0.638
IMP_{PLS}	6.062 [5.233, 6.891]	**0**	6.271 [5.648, 6.893]	**0**	−0.212 [−0.912, 0.487]	0.521
IMP_{ARS}	5.791 [5.316, 6.266]	**0**	5.978 [5.511, 6.445]	**0**	−0.187 [−0.852, 0.478]	0.551
IMP_{DOM}	6.393 [4.829, 7.957]	**0**	6.285 [5.681, 6.89]	**0**	0.108 [−1.321, 1.537]	0.872
IMP_{LIK}	6.518 [5.2, 7.837]	**0**	6.689 [6.193, 7.185]	**0**	−0.167 [−1.685, 1.352]	0.815
TES_{PLS}	5.675 [4.885, 6.465]	**0**	5.939 [5.407, 6.471]	**0**	−0.262 [−1.396, 0.871]	0.623
TES_{ARS}	5.479 [4.924, 6.033]	**0**	5.639 [5.088, 6.189]	**0**	−0.161 [−0.768, 0.447]	0.575
TES_{DOM}	6.152 [5.416, 6.888]	**0**	6.515 [5.783, 7.247]	**0**	−0.363 [−1.114, 0.387]	0.313
TES_{LIK}	5.965 [5.276, 6.654]	**0**	6.345 [5.727, 6.963]	**0**	−0.376 [−1.574, 0.822]	0.507
REF_{PLS}	5.908 [5.042, 6.773]	**0**	5.86 [5.133, 6.587]	**0**	0.049 [−0.61, 0.707]	0.873
REF_{ARS}	5.873 [4.925, 6.82]	**0**	5.676 [4.75, 6.602]	**0**	0.197 [−0.553, 0.947]	0.572
REF_{DOM}	6.392 [5.605, 7.179]	**0**	6.206 [5.711, 6.702]	**0**	0.183 [−0.579, 0.945]	0.605
REF_{LIK}	6.163 [5.249, 7.077]	**0**	6.069 [5.572, 6.565]	**0**	0.092 [−0.773, 0.957]	0.817

Table 5
IPD-S results for the dependent variables involved in the study of the experiment- and participant-level moderators (*i.e.*, Graduate, Java Programming, Professional Developer, and Unit Testing). p-values less than $\alpha = 0.1$ are highlighted in bold.

Dep. var.	Graduate		Java programming		Professional developer		Unit testing	
	Estimate [95% CI]	p-value	Estimate [95% CI]	p-value	Estimate [95% CI]	p-value	Estimate [95% CI]	p-value
APP_{LIK}	2.015 [−0.747, 4.776]	0.137	0.023 [−0.025, 0.07]	0.319	0.039 [−0.150, 0.227]	0.662	−0.278 [−0.512, −0.043]	**0.021**
IMP_{DOM}	1.577 [−0.645, 3.799]	0.147	0.009 [−0.032, 0.05]	0.644	0.015 [−0.126, 0.155]	0.821	−0.069 [−0.280, 0.141]	0.515
IMP_{LIK}	1.676 [−0.947, 4.3]	0.187	0.019 [−0.024 0.062]	0.356	0.054 [−0.082, 0.191]	0.4	−0.138 [−0.362, 0.085]	0.222
TES_{PLS}	1.057 [−1.61, 3.724]	0.402	0.028 [−0.015, 0.07]	0.179	−0.051 [−0.186, 0.083]	0.419	−0.246 [−0.456, −0.037]	**0.021**
TES_{LIK}	1.519 [0.173, 2.864]	**0.03**	0.029 [−0.013, 0.071]	0.154	0.105 [−0.037, 0.247]	0.133	−0.253 [−0.467, −0.039]	**0.021**

with the Approach × Graduate interaction term to complement the study of the experiment-level moderator (Activity 4.1). For the above-mentioned variables, we then fitted LMMs, each with an interaction term, to identify participant-level moderators (Activity 4.2). The considered interactions are Approach × {*Java Programming*, *Professional Developer*, *Unit Testing*}. These three factors represent, respectively, the months of experience as java programmer, as professional developer, and in unit testing. The LMM results about the study of the experiment- and participant-level moderators are summarized in Table 5. In this subsection, we present the results of our exploratory analyses by dependent variable.

4.4.1. Exploratory analysis of APP_{LIK}

As Fig. 6 shows, the heterogeneous results for APP_{LIK} should not be due to an experiment-level moderator because a significant (Q-test p-value = 0.036) and considerable ($I^2 = 77\%$) between-study heterogeneity can be observed within the undergraduate subgroup of experiments. Also, there is no significant difference between TDD and YW when considering the levels of the Graduate factor—the CIs of the joint effects span 0 for both subgroups of experiments. The IPD-S results in Table 5 do not show a significant effect of the Approach × Graduate interaction—the p-value (0.137) is not less than 0.1. Therefore, to explain the heterogeneous results on APP_{LIK}, we need to focus on participant-level moderators. As shown in Table 5, the highest estimate (in absolute value) is the one for the Approach × Unit Testing interaction (−0.278) and it is also significant (p-value = 0.021). That is, the moderator that better explains the heterogeneous results on APP_{LIK} is Unit Testing (*i.e.*, the months of experience with unit testing). The results from the LMM fitted for APP_{LIK} suggest that the higher the months of experience with unit testing, the more negative the affective reactions to TDD, in terms of liking, are (as compared to YW).

4.4.2. Exploratory analysis of IMP_{DOM}

As shown in Fig. 6, the between-study heterogeneity for IMP_{DOM} is not significant within both subgroups of experiments (as the Q-test p-values suggest). Moreover, the joint effect for the undergraduate subgroup is not significant, while the one for the Graduate factor is significant. Accordingly, the Graduate moderator can explain the heterogeneous results on IMP_{DOM} although the IPD-S results in Table 5 fail to show a significant effect of the Approach × Graduate interaction. When considering the participant-level moderators (Table 5), we can observe that all estimate p-values are not less than 0.1. Moreover, the estimates are close to 0 for any interaction. We thus confirm with more strength that the Graduate moderator can explain the heterogeneous results on IMP_{DOM}. In particular, graduates who applied TDD were more in-control than those who applied YW when implementing production code, while for undergraduates this was not true.

4.4.3. Exploratory analysis of IMP_{LIK}

By looking at Fig. 6, we can observe that there is a significant and substantial between-study heterogeneity for IMP_{LIK} when considering the undergraduate subgroup. Moreover, the results in Table 5 suggest that the Graduate factor is not significant. Therefore, the heterogeneous results (highlighted in Section 4.3.2) should be due to a participant-level moderator. However, Table 5 does not show significant interaction effects for the participant-level moderators—all p-values are not less than 0. The highest estimate (in absolute value) is the one for the Approach × Unit Testing interaction (−0.138). Therefore, the moderator, better explaining the heterogeneous results on IMP_{LIK}, seems to be Unit Testing. The higher the months of experience with unit testing, the less the participants liked implementing production code when using TDD (as compared to YW).

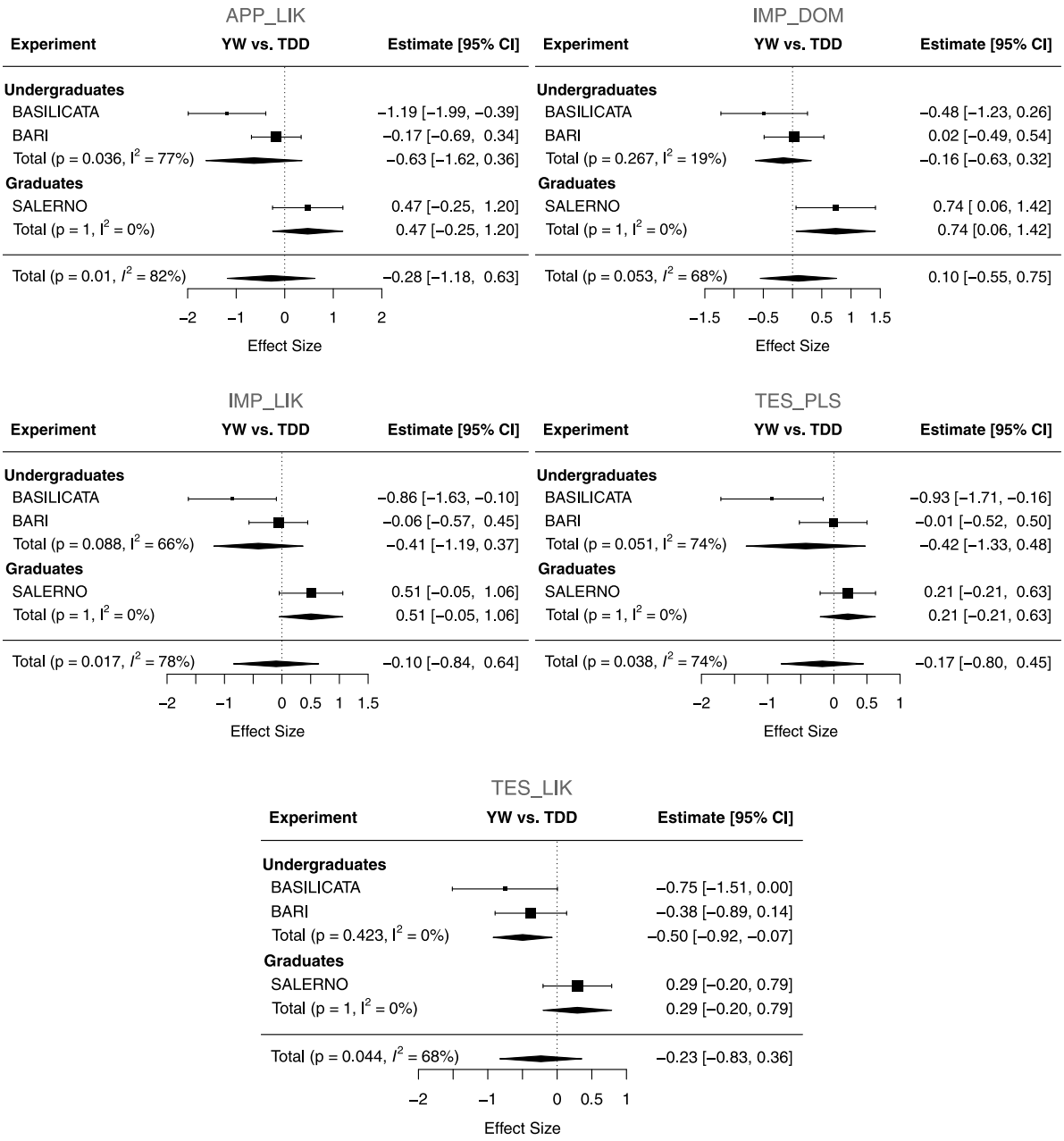


Fig. 6. Forrest plots depicting the SMD values for the dependent variables involved in the study of the Graduate experiment-level moderator.

4.4.4. Exploratory analysis of TES_{PLS}

The Q-test p-value and I^2 value in Fig. 6 indicates that there is a significant and substantial between-study heterogeneity for TES_{PLS} within the undergraduate subgroup. The effect of the Approach × Graduate interaction is not significant as the results reported in Table 5 suggest. Accordingly, the participant-level moderators should be investigated, rather than the experiment-level moderation, to explain the heterogeneous results for TES_{PLS} across the experiments. The IPD-S results in Table 5 show a significant effect for Approach × Unit Testing (p-value = 0.021) with an estimate equal to −0.246—it is the highest obtained estimate (in absolute value). Based on these results, the Unit Testing moderator explains the heterogeneous results for TES_{PLS}. In particular, the higher the months of experience with unit testing, the less happy the participants were when testing production code using TDD (as compared to YW).

4.4.5. Exploratory analysis of TES_{LIK}

As Fig. 6 shows, there is not a significant between-study heterogeneity for TES_{LIK} in both subgroups of experiments. Moreover, the joint effect for the undergraduate subgroup is significant and medium (SMD = −0.50). The IPD-S results in Table 5 show a significant effect of the Approach × Graduate interaction. Therefore, the Graduate moderator can explain the heterogeneous results for TES_{LIK}. As for the participant-level moderators, we can observe that there is a significant effect for Approach × Unit Testing (p-value = 0.021). The estimate is equal to −0.253. Therefore, besides the Graduate moderator, the Unit Testing moderator can explain the heterogeneous results for TES_{LIK}. In particular, the undergraduates who applied TDD liked testing production code less than the undergraduates who applied YW. For the graduates, this was not true. Also, the higher the months of experience with unit testing, the less the participants liked testing production code when using TDD (as compared to YW).

Table 6
p-values for the Approach factor that the ATS method returned for the Line Coverage, Mutation Score, and Tests dependent variable in each experiment. *p*-values less than $\alpha = 0.05$ are highlighted in bold.

Dep. var.	BASILICATA		BARI		SALERNO	
	p-value	$H0_X$ outcome	*p*-value	$H0_X$ outcome	*p*-value	$H0_X$ outcome
Line Coverage	0.569	Failed to reject	0.376	Failed to reject	0.69	Failed to reject
Mutation Score	0.897	Failed to reject	0.708	Failed to reject	1	Failed to reject
Tests	0.435	Failed to reject	0.315	Failed to reject	0.108	Failed to reject

4.5. Further analyses

In this subsection, we present the results of a quantitative further analysis. In particular, besides investigating whether and in which phases TDD influences the affective states of developers, we also studied whether TDD affects the code quality of the developed software, as well as the number of written tests. To quantify the code quality of the developed software, we used two (dependent) variables: Line Coverage and Mutation Score. The former represents the percentage of lines (of production code) that are covered by the test suite each participant wrote. The latter represents the percentage of mutants that are killed by the participant's test suite.[8] We computed both Line Coverage and Mutation Score by using PIT (Coles et al., 2016), a state-of-the-art mutation testing tool for Java (Kintis et al., 2018). To generate the mutants and thus compute the mutation score, we used the default mutation operators of PIT since they are designed to be stable (*i.e.,* not too easy to detect) and minimize the number of equivalent mutants (*i.e.,* mutants functionally equivalent to the original program). Both Line Coverage and Mutation Score are expressed in percentage values—the higher the values of these variables, the better it is. As for the number of written tests, we used Tests as the dependent variable. This variable indicates the number of tests written by a given participant and therefore it assumes integer values greater than or equal to zero.

In Fig. 7, we show the boxplots depicting the values for the Line Coverage, Mutation Score, and Tests dependent variables for each approach in each experiment. As for Line Coverage, we can observe that there is not a huge difference between TDD and YW since, whatever the experiment is, the boxes overlap one another. Similar outcomes can be observed when considering Mutation Score and Tests.

To confirm these outcomes, we also performed inferential analyses. In particular, we tested the $H0_{LineCoverage}$, $H0_{MutationScore}$, and $H0_{Tests}$ null hypotheses – *i.e., there is no significant difference between TDD and YW with respect to Line Coverage (or Mutation Score or Tests, respectively)* – by using the ATS method (*i.e.,* we applied the same analysis procedure of Activity 2.3 shown in Section 3.8). The results from the inferential analyses are shown in Table 6; where we can observe that there is no significant difference between TDD and YW for Line Coverage, Mutation Score, and Tests in any experiment.

5. Discussion and limitations of experiment results

In this section, we discuss the results from the experiments and highlight implications from these results from the educator, researcher, and practitioner perspectives. We conclude by discussing possible limitations of these results.

[8] In mutation testing, the faulty version of a program, obtained by applying a mutation operator that automatically seeds a mutation fault in that program, is called mutant. A mutant is said killed when a test case of the test suite detects the corresponding mutation fault.

5.1. Discussion of results

Replications that do not draw the same conclusions as the baseline experiment can be viewed as successful on a par with replications that come to the same conclusions as the baseline experiment (Shull et al., 2008). Our replications fall into the former case since the outcomes of the replicated experiments, taken individually, do not fully confirm the outcomes of the baseline experiment. In particular, we observed that the participants in the baseline experiment *(i)* liked TDD significantly less as compared to a non-TDD approach; *(ii)* liked implementing production code with TDD significantly less; and *(iii)* were significantly less happy when testing production code using TDD. The replications cannot support these findings because, as shown in Section 4.2, either we did not observe any significant difference between TDD and YW (this is the case of the first replication) or we observed significant differences that did not emerge in the baseline experiment (this is the case of the second replication). Neither aggregating the results from all experiments allowed showing the same significant differences as observed in the baseline experiment (see Section 4.3). Accordingly, some moderators had to have modulated the effect of the used development approach on the affective reactions of the participants for $\texttt{APP}_{\texttt{LIK}}$, $\texttt{IMP}_{\texttt{LIK}}$, and $\texttt{TES}_{\texttt{PLS}}$—*i.e.,* the dependent variable for which the results from the baseline experiment are not confirmed by the results of the replications. The results from our exploratory analyses (see Section 4.4) indicate that the moderator that better explained the above-mentioned inconsistent/heterogeneous results (*i.e.,* as far as $\texttt{APP}_{\texttt{LIK}}$, $\texttt{IMP}_{\texttt{LIK}}$, and $\texttt{TES}_{\texttt{PLS}}$ is concerned) across the experiments is the experience with unit testing. In particular, the higher the months of experience with unit testing, the more negative the affective reactions, due to TDD, are. Since the participants in any experiment did not know TDD (before being recruited for participating in the experiments), they were therefore used to practice unit testing in a test-last manner. In other words, the participants were used to write unit tests after they had written production code—in contrast to TDD, where unit tests are written before producing code. That is to say that the participants in the baseline experiment were probably more conservative and less prone to change the order with which they usually wrote production and testing code. Therefore, the affective reactions due to TDD of the participants in the baseline experiment were more negative as compared to the affective reactions of the participants in the two replications.

The above-mentioned finding (*i.e.,* the higher the experience with unit testing, practiced in a test-last manner, the more negative the affective reactions caused by TDD) can be of interest to CS educators who teach unit testing. For example, they should start teaching TDD as soon as possible to lessen the negative affective reactions that TDD might cause since there is empirical evidence showing that, with time, TDD leads developers to write more unit tests with a higher fault-detection capability (Fucci et al., 2018; Baldassarre et al., 2021). As far as researchers is concerned, they could be interested in studying whether the experience with unit testing plays a relevant role in the relationship between TDD and the affective reactions of developers (as far as $\texttt{APP}_{\texttt{LIK}}$, $\texttt{IMP}_{\texttt{LIK}}$, and $\texttt{TES}_{\texttt{PLS}}$ is concerned) through experiments specifically designed

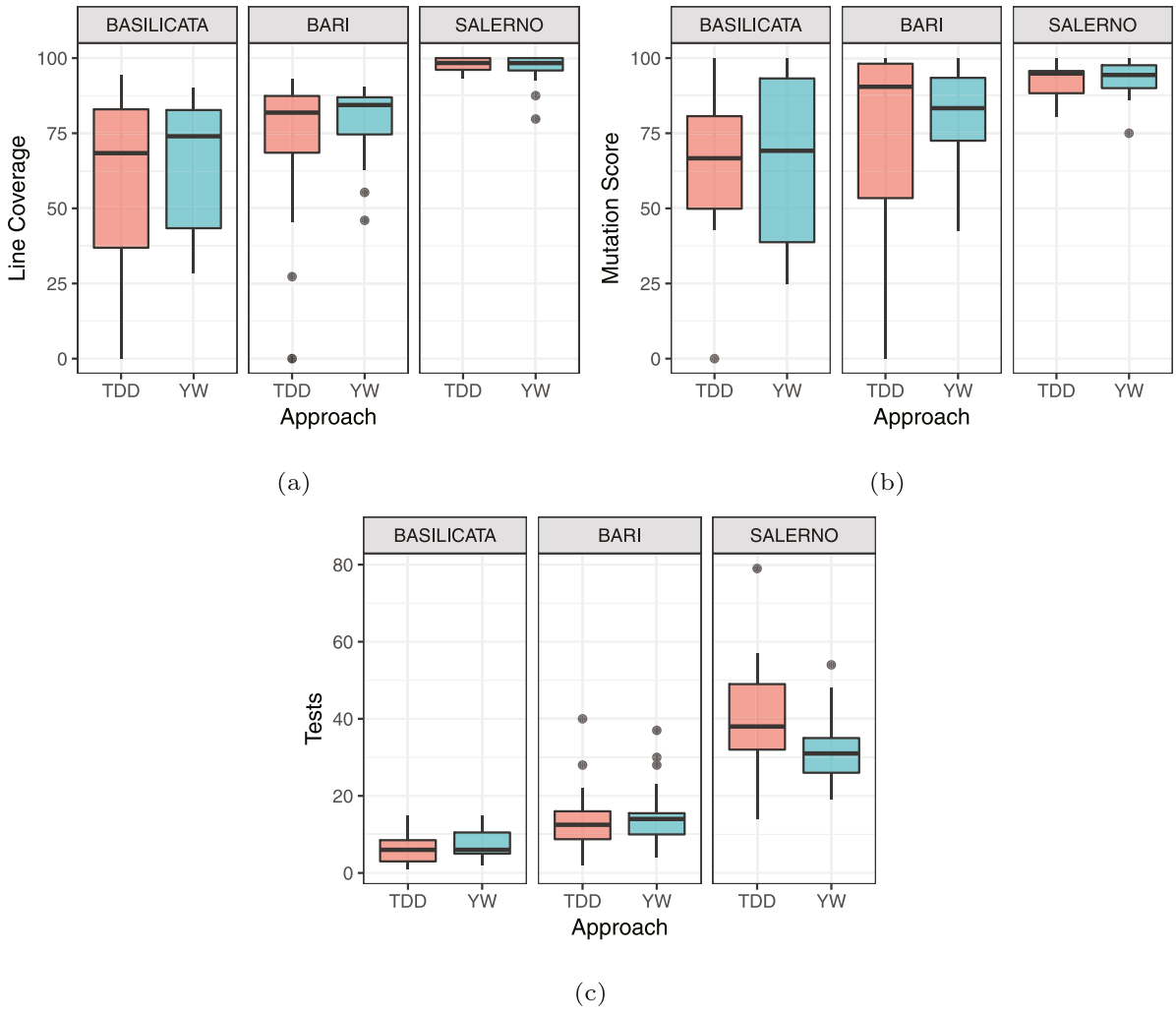


Fig. 7. Boxplots summarizing the values for the Line Coverage (a), Mutation Score (b), and Tests (c) dependent variables for each approach in each experiment.

for such a purpose. In particular, researchers could conduct experiments with a 2 × 2 factorial design (Wohlin et al., 2012) where the two factors are the development approach and the experience with unit testing.

Current research on the claimed effects of TDD (*i.e.,* better internal software quality and increased developers' productivity) shows inconclusive results (*e.g.,* Turhan et al., 2010; Karac and Turhan, 2018), which can be attributed to the disliking that developers experience when using TDD (*e.g.,* Erdogmus et al., 2010). The results from the baseline experiment give credit to such a postulation, despite the results from the replications do not (due to the less experience with unit testing of the participants). The question that now arises is whether the disliking that developers experience when using TDD can change over time (*e.g.,* in relation to the improved experience with TDD). Researchers could conduct observational studies with a cohort of developers to answer such a question. Also, the disliking that the participants in the baseline experiments experienced when using TDD could explain why TDD is not widely practiced in open-source projects (Beller et al., 2019; Barletta et al., 2020).

As for other investigated constructs (*i.e.,* those measured through: $\texttt{APP}_{\texttt{PLS}}$, $\texttt{APP}_{\texttt{ARS}}$, $\texttt{APP}_{\texttt{DOM}}$, $\texttt{IMP}_{\texttt{PLS}}$, $\texttt{IMP}_{\texttt{ARS}}$, $\texttt{TES}_{\texttt{ARS}}$, $\texttt{TES}_{\texttt{DOM}}$, $\texttt{REF}_{\texttt{PLS}}$, $\texttt{REF}_{\texttt{ARS}}$, $\texttt{REF}_{\texttt{DOM}}$, and $\texttt{REF}_{\texttt{LIK}}$), the outcomes of the baseline experiment are confirmed by the replications and when analyzing the results jointly. Also, we did not observe any significant between-study heterogeneity. In other words, the statistical conclusion for each of these constructs is that we failed to reject the corresponding null hypothesis. Such a kind of result is known as a *negative result*. Negative results are important in showing research directions that did not pay off and help in reorienting the research efforts (Paige et al., 2017). Accordingly, our results suggest avoiding spending effort investigating the constructs for which we did not obverse any significant difference and did not observe any significant between-study heterogeneity. On the other hand, further replications with other types of participants (*e.g.,* participants with high experience in unit testing) are necessary to investigate the constructs for which we obtained heterogeneous results.

When considering the subgroup of experiments with undergraduates (*i.e.,* the baseline experiment and first replication), we observed that the participants using TDD liked testing production code significantly less (see Section 4.4). Previous work (Romano et al., 2016, 2017) shows that TDD developers create a mental model of the task solution, which is then translated into unit tests. Undergraduates can be uncomfortable with such an activity due to the counter-intuitiveness of the TDD testing phase, along with the difficulty of writing tests of good granularity in the absence of the underlying production code (Fucci et al., 2017; Karac and Turhan, 2018). Conversely, undergraduates following a non-TDD approach can decide when and what to test without (mindlessly) following a process. Such freedom of action – *e.g.,* testing what is worth according to developer's understanding – can explain the higher values for $\texttt{TES}_{\texttt{LIK}}$ among the undergraduates using a non-TDD approach. Longitudinal studies of TDD developers' affective

states can be also useful in this case. The results regarding the negative affective reactions of undergraduates to testing code suggest that, for greenfield development tasks, undergraduates can skip TDD for few initial iterations and rely on their preferred development approach. This should positively impact their motivation and job satisfaction (Turhan et al., 2010; Graziotin et al., 2018), along with productivity (Graziotin et al., 2013). Such a finding is clearly relevant for software managers interested to have motivated developers in a software project, but also for researchers that could further study on the relations between motivation and job satisfaction and the use of TDD.

5.2. Threats to validity

To understand both strengths and limitations of our experimental results, we discuss the threats that could affect their validity by using Wohlin et al.'s guidelines (Wohlin et al., 2012). We ranked the threats from the most to the least sensible for our research goal—*i.e.,* from internal to external validity threats. The rationale behind this decision relies on the fact that we were more interested in studying that cause–effect relationships were correctly identified in our experiments.

Threats to internal validity. A *selection* threat might have affected our results since the participation in the experiments was voluntary—the participants in the experiments might be more motivated than actual developers. Another threat that might have affected our results is *resentful demoralization*, namely participants assigned to a less desirable treatment might not behave as they normally would. This kind of threat holds for BASILICATA and BARI, but not for SALERNO where the participants experimented both TDD and YW. To deal with a *threat of diffusion or treatments imitations*, we monitored the participants to prevent them from exchanging information during the experimental sessions. Exploratory analyses, like ours, are unable to demonstrate causation (Santos et al., 2019) (threat of *ambiguity about direction of causal influence*) although they can prove associations. This is why we suggested (in Section 5) conducting experiments with a 2×2 factorial design to better study the role that the experience with unit testing has in the relationship between TDD and the affective reactions of developers.

Threats to construct validity. We measured each dependent variable once by using a self-assessment instrument (*i.e.,* SAM). As so, in case of measurement bias, this might affect the obtained results (threat of *mono-method bias*). Although we did not disclose the research goals of our study to the participants, they might have guessed them and changed their behavior based on their guess (threat of *hypotheses guessing*). This threat should be more relevant for SALERNO since the participants experimented both YW and TDD. We did not evaluate the participants in the experiments to mitigate a threat of *evaluation apprehension*. In particular, we informed the participants in all experiments that they would get two bonus points on the final exam mark regardless of their performance. There might be a threat of *restricted generalizability across constructs*. To deal with such a threat, we analyzed the code coverage, mutation score, and number of written tests when using TDD or YW, and observed no significant difference. The results of this further analysis are reported in Section 4.5. However, the approach might affect other non-measured constructs (*e.g.,* cognitive load). The exploratory analyses pose a threat of *confounding constructs and levels of constructs*. In other words, despite we studied four moderators, other confounding variables might also be responsible for the heterogeneous results.

Threats to conclusion validity. We mitigated a threat of *random heterogeneity of participants* through two countermeasures: *(i)* we involved students so having samples of participates, in each experiment, with background and skills as much similar as possible; and *(ii)* the participants in each experiment underwent a training to make them as more homogeneous as possible within the groups. It is worth mentioning that the training of the participants across the experiments was similar even if organized in different fashion due to the different peculiarity of the experimental procedure applied in the experiments (Section 3.7). A threat of *reliability of treatment implementation* might have occurred in the experiments. For example, a few participants might have followed TDD more strictly than others and this might have influenced their affective reactions. To mitigate this threat, we reminded the participants assigned to TDD to apply such a development approach as more strictly as possible on several occasions during the experiments. It is worth noting that not checking the conformance to the TDD process is common in primary studies like ours (Santos et al., 2021). Although SAM is one of the most reliable instruments for measuring affective reactions (Morris et al., 2002), there might be a threat of *reliability of measures*—the measures gathered using SAM, as well as the liking scale, are subjective in nature. Also, in TDD, the testing, implementation, and refactoring phases are intertwined with one another. Therefore, it could be not easy, for the participants using TDD, to clearly separate these phases and then rate their affective reactions to these phases. On the other hand, for the participants using YW, the testing, implementation, and refactoring phases can be seen as distinct steps. This difference between TDD and YW might have affected how the participants rated their affective state to testing, implementing, and refactoring code. According to Santos et al. (2019), exploratory analyses (like ours) might lead to spurious significant results because they require, for each dependent variable, to fit a LMM per moderator (threat of *error rate*). A threat of *violated assumptions of statistical tests* might be present since, as done by Santos et al. (2019), we did not check the normality assumption of LMMs (Whitehead, 2003). However, LMMs are robust to departures from normality (Fagerland, 2012; McCulloch and Neuhaus, 2011)—especially when the sample size is large as it happens when the raw data of different experiments are pooled together (Lumley et al., 2002) (as it is our case). Moreover, we mostly used LMMs in tandem with meta-analyses of effect sizes so mitigating such a threat.

Threats to external validity. We involved graduates and undergraduates in CS. This implies that generalizing the results to the population of professional developers might lead to a threat of *interaction of selection and treatment*. However, the use of students has the advantage that they have a more homogeneous background and are particularly suitable to obtain preliminary evidence from empirical studies (Carver et al., 2003). Therefore, the use of students could be considered appropriate, as suggested in the literature (Carver et al., 2003; Höst et al., 2000). The use of BSK and MR might represent a threat of *interaction of setting and treatment*. These experimental objects are commonly used in empirical studies on TDD (*e.g.,* Fucci et al., 2016a, 2018, 2016b) and can be fulfilled in about three hours (Fucci et al., 2018) so allowing a better control over the participants. Furthermore, both BSK and MR can be implemented without using mocking techniques (*e.g.,* mock objects or spies) and frameworks; and the participants did not (and were not allowed to) use any mocking technique or framework. This design choice was to limit a threat of *interaction of different treatments* (*i.e.,* TDD and mocking). While mitigating such a threat, we acknowledge a threat of *ecological validity* since TDD and mocking are sometime used together in the industrial practice.

6. Survey

We conducted an explanatory survey to explain the results of the experiments and complement these results by performing

a *methodological triangulation* (Thurmond, 2001). In particular, we took a qualitative perspective by involving respondents with TDD experience to explain some heterogeneous results across the experiments. Moreover, we collected their perspective on which affective states developers experience due to TDD.

6.1. Goals

In the survey, we studied the two following RQs:

RQ4. Does the experience with unit testing, applied in a test-last manner, lead to negative affective reactions towards TDD?

RQ5. Which affective states do developers experience when using TDD?

The goal of RQ4 is to confirm (or refute) that the higher the experience with unit testing (applied in a test-last manner), the more negative the affective reactions caused by TDD are. As for RQ5, we wanted to understand and possibly explain which affective states TDD provokes by considering the perspectives of professionals and researchers with TDD experience. The answer to RQ5 could help us to better understand TDD as a phenomenon from a new perspective and derive suggestions to deal with specific affective states.

6.2. Respondents

The survey respondents were 11 TDD lecturers, developers, and/or researchers who attended the XP2020 conference. The participation in the survey was voluntary because we wanted to have motivated respondents who were likely to answer the questions truthfully. The XP2020 Onsite-research Track organizers encouraged the conference attendees to fill in our survey by using both social media and conference website—a video was used to sketch our investigation so encouraging the participation in the study.[9] It is easy to follow that we did not formally invite XP2020 attendees to fill in the survey—this is why we do not have a response rate.

6.3. Data collection method and execution

The survey was questionnaire-based (Wohlin et al., 2012) and consisted of both open- and closed-ended questions. The survey questionnaire was introduced with a brief motivation sketching the general problem of the investigation. Furthermore, we clarified that any information gathered through the questionnaire was considered confidential and the data would be used for research purposes and shared in an anonymous form only.

We identified three areas of interest that we wanted to gather information about. Therefore, we organized the questionnaire into three parts. The first part focused on demographic information about the respondents (*e.g.,* job title, expertise with TDD, *etc.*.) to characterize our sample. The second and third parts were conceived to answer RQ4 and RQ5, respectively. In particular, in the second part, we included questions on the respondents' opinion about the relationship between the experience with unit-last testing, applied in a test-last manner, and the affective reactions to TDD. In the last part of the questionnaire, we included an open question to ask the respondents their thought on the affective states (*e.g.,* happiness, frustration, *etc.*.) that TDD might provoke (*e.g.,* personal/friend's anecdote). We developed the survey questionnaire following Ciolkowski et al.'s schema (Ciolkowski et al., 2003). We used Google forms to create the survey questionnaire, share it with the respondents, and gather the responses.

[9] www.agilealliance.org/xp2020/xp-2020-online-program/onsite-research-track

7. Survey results

In this section, we first present the demographic information results, and then the results regarding the study of RQ4 and RQ5.

7.1. Demographic information

In Fig. 8 (upper part), we summarize the answers to the questions about the demographic part of the survey (*i.e.,* Questions 1.1 to 1.9).

The age of the respondents ranges from 25 to 68 years with an average age of about 44 years. The female respondents were three, while the male respondents eight. They came from nine different countries—the most represented countries were the USA and the UK with two occurrences each. Most respondents had a Master's or Doctorate degree (nine). As for their job title, the most represented classes were software consultants, followed by Ph.D. students and assistant professors. The expertise of the respondents with TDD was, overall, high—eight out of 11 respondents stated that they were experts of TDD or had an advanced experience with TDD. Most respondents have been practicing TDD for a long time—only two respondents had been practicing TDD for less than one year. The context in which they practiced TDD the most was teaching (ten). The respondents also used TDD in industrial (six), research (four), and open-source (four) contexts.

7.2. RQ4—Experience with unit testing and affective reaction towards TDD

In Fig. 8 (lower part), we summarize the answers to the questions for RQ4 (*i.e.,* Questions 2.1 to 2.3). It is worth noting that such questions were related to the results from the baseline experiment not confirmed by the results from the first replications (BARI)—*i.e.,* the results regarding $\texttt{APP}_{\texttt{LIK}}$, $\texttt{IMP}_{\texttt{LIK}}$, and $\texttt{TES}_{\texttt{PLS}}$. This is because we conducted the survey in parallel with the second replication; therefore, we did not know that we would have observed heterogeneous results for $\texttt{APP}_{\texttt{DOM}}$ and $\texttt{TES}_{\texttt{LIK}}$ when analyzing the data from the three experiments jointly. We devise this point as a possible direction for future work.

As for Question 2.1, we can observe a slight trend towards the agreement that having much experience with unit testing causes more dislike towards TDD than having no/little experience. It seems that there is not a clear trend on Question 2.2. Finally, most respondents (five) thought that having much experience with unit testing makes developers more unhappy in the testing phase of TDD than having no/little experience (Question 2.3).

7.3. RQ5—Affective states when using TDD?

Based on respondents' responses/opinions, the testing phase of TDD is the one that causes more negative affective states to the developer. The opinion of R2 (*i.e.,* the second respondent) about this point follows: *"TDD in the red phase causes more anxiety and frustration"*. A possible reason seems to be the counter-intuitive and hard-to-tackle test-first order that characterizes TDD. For example, R9 stated: *"I have found that Red, fail first, is counter-intuitive for many developers and the more experienced the more resistance"*. Also, R11 stated: *"The test step is really hard because you have to find the right tests and the right order to tackle them"*.

Negative affective states, like unhappiness, can be also experienced when the TDD developer cannot immediately write production code to make unit tests pass or when she writes code that suddenly makes a lot of tests fail. R8's perspective follows: *"If you (a) don't immediately write the correct code to make the test*

1.1. How old are you?					
Min: 25; Median: 43; Mean: 44.09; Max: 68					
1.2. What is your gender?					
Male	8	Female	3	Prefer not to answer	0
1.3. What country are you from?					
United States	2	United Kingdom	2	Brazil	1
India	1	Iran	1	Netherlands	1
Portugal	1	Germany	1	Sweden	1
1.4. What is your education level?					
Associate degree	1	Bachelor's degree	1	Master's degree	5
Doctorate degree	4				
1.5. What is your current job title?					
Software consultant	4	Team lead	1	PhD student	2
Assistant professor	2	Associate professor	1	Reader	1
1.6. How would you rate your expertise with TDD?					
Fundamental Awareness	0	Novice	1	Intermediate	2
Advanced	5	Expert	3		
1.7 How long have you been practicing TDD?					
From less than 1 year	2	From 1 to 5 years	1	From 6 to 10 years	5
From more than 10 years	3				
1.8. While coding, to what extent do you practice TDD?					
Never	0	Occasionally	2	Sometimes	1
Often	8	Always	0		
1.9. In which context do you practice TDD?					
Teaching context	10	Research context	4	Industrial context	6
Open-source context	4				
2.1. Having much experience with unit testing (applied in a test-last manner) causes more dislike towards TDD than having no/little experience.					
Strongly disagree	0	Disagree	2	Neutral	4
Agree	4	Strongly agree	1	Do not know	0
2.2. Having much experience with unit testing (applied in a test-last manner) causes more dislike towards the implementation (Green) phase of TDD than having no/little experience.					
Strongly disagree	0	Disagree	3	Neutral	1
Agree	3	Strongly agree	1	Do not know	3
2.3. Having much experience with unit testing (applied in a test-last manner) makes developers more unhappy in the testing (Red) phase of TDD than having no/little experience.					
Strongly disagree	0	Disagree	1	Neutral	2
Agree	5	Strongly agree	0	Do not know	3

Fig. 8. Summary of the results regarding the first two parts of the survey.

pass, or (b) write code that suddenly makes a lot of tests fail, you get more unhappy and more excited and not in-control".

The phase underlying TDD that mostly causes neutral affective states should be the refactor one. On this point, R8 wrote: *"Refactoring as such is less emotional".*

8. Discussion and limitations of survey results

In this section, we first discuss the survey results and then the limitations for these results.

8.1. Discussion of results

The survey results help us to confirm the exploratory analysis results. In particular, the latter results suggest that the moderator that better explains the inconsistent/heterogeneous results across the experiments – for APP_{LIK}, IMP_{LIK}, and TES_{PLS}– is the developers' experience with unit testing, applied in a test-last manner. The survey respondents confirmed that having more experience with unit testing, applied in a test-last manner, causes more dislike towards TDD and makes developers using TDD more unhappy when testing production code.

When considering the subgroup of experiments with undergraduates, we observed that the participants using TDD liked significantly less testing production code. This result seems to be confirmed in the survey since TDD causes negative affective states when testing code. In turn, this can explain the inconclusive results of the research on the claimed effects of TDD (Erdogmus et al., 2010).

8.2. Threats to validity

In this section, we discuss the threats to validity that could affect the survey results. The discussion of these threats is shown in increasing priority order (*i.e.,* internal, external, and construct).

Threats to internal validity. In surveys, it is impossible to know whether respondents answer truthfully. To mitigate such a threat, the participation in our survey was voluntary basis. Another potential threat regards the comprehension of the survey questions. To lessen this threat, we asked a researcher, not involved in the study, to read and provide possible issues concerned with the comprehension of the survey questions before the study took place—only minor changes were needed.

Threats to external validity. This kind of threat concerns the representativeness of the respondents. The reader might object that XP2020 attendees were not representative of the population from which we would like to gather feedback to explain the results from the experiments. We believe that the attendees of that conference are experienced in the application and/or teaching of TDD, so suitable for participating in the survey—the demographic information reassures us on this matter (see Section 7.1). XP2020 was held virtually so possibly affecting the number of people who could participate in the survey. We believe that, in a traditional context, we would have had more respondents since it would have been easier to personally contact potential respondents during coffee breaks and social moments. However, the respondents in our sample should be very motivated. Finally, we acknowledge that the views of XP2020 attendees may not necessarily reflect or accurately explain the feelings or reasoning of the participants in the experiments.

Threats to construct validity. This kind of threat concerns the questions and scales used to assess the constructs. We designed our survey by using a standard approach and scales (Ciolkowski et al., 2003). The design of our study also underwent a peer-review process—*i.e.,* it involved the PC members of the XP2020 Onsite-research Track.

9. Final remarks

We studied whether, and in which phases, TDD influences the affective states of developers in terms of pleasure, arousal, dominance, and liking. We conducted a baseline experiment and two replications with participants from different contexts. We analyzed the data from these experiments both individually and jointly, and performed a survey with people having experience with TDD to complement and explain the results from the experiments. The experiments and survey can individually provide limited insights, while the combination of their outcomes can provide a substantial contribution towards understanding the affective reactions of developers when using TDD. The most important conclusions of the research presented in this paper are:

- The experience of developers with unit testing, applied in a test-last manner, seems to be the most important moderator, which: *(i)* causes more dislike towards TDD (as compared to a non-TDD approach); *(ii)* causes more dislike when implementing production code with TDD; and *(iii)* makes developers using TDD more unhappy when testing production code. In particular, the higher the experience with unit testing applied in a test-last manner, the more negative the affective reactions due to TDD. The results of our survey provide further support for findings *(i)* and *(iii)*.
- We observed that, when considering the subgroup of experiments with undergraduates, the participants using TDD liked significantly less testing production code. This seems to be confirmed by the survey results.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

For this work, Davide Fucci would like to acknowledge the support of the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology.

References

Baldassarre, M.T., Caivano, D., Fucci, D., Juristo, N., Romano, S., Scanniello, G., Turhan, B., 2021. Studying test-driven development and its retainment over a six-month time span. J. Syst. Softw. 110937.

Barletta, V.S., Caivano, D., Dimauro, G., Nannavecchia, A., Scalera, M., 2020. Managing a smart city integrated model through smart program management. Appl. Sci. 10 (2), http://dx.doi.org/10.3390/app10020714.

Beck, K., 2003. Test-Driven Development: By Example. Addison-Wesley.

Beller, M., Gousios, G., Panichella, A., Proksch, S., Amann, S., Zaidman, A., 2019. Developer testing in the IDE: Patterns, beliefs, and behavior. IEEE Trans. Softw. Eng. 45 (3), 261–284.

Borenstein, M., Hedges, L., Higgins, J., Rothstein, H., 2009. An Introduction to Meta-Analysis. John Wiley & Sons.

Bradley, M.M., Lang, P.J., 1994. Measuring emotion: The self-assessment manikin and the semantic differential. J. Behav. Ther. Exp. Psychiatry 25 (1), 49–59.

Brunner, E., Konietschke, F., Pauly, M., Puri, M.L., 2017. Rank-based procedures in factorial designs: Hypotheses about non-parametric treatment effects. J. R. Statist. Soc. Ser. B Stat. Methodol. 79 (5), 1463–1485.

Carver, J., Jaccheri, L., Morasca, S., Shull, F., 2003. Issues in using students in empirical studies in software engineering education. In: Proceedings of International Symposium on Software Metrics. IEEE, pp. 239–249.

Ciolkowski, M., Laitenberger, O., Biffl, S., 2003. Software reviews: The state of the practice. IEEE Softw. 20 (6), 46–51.

Cohen, J., 1992. A power primer.. Psychol. Bull. 112, 155–159.

Coles, H., Laurent, T., Henard, C., Papadakis, M., Ventresque, A., 2016. PIT: A practical mutation testing tool for Java (Demo). In: Proceedings of International Symposium on Software Testing and Analysis. ACM, pp. 449–452.

Curumsing, M.K., Fernando, N., Abdelrazek, M., Vasa, R., Mouzakis, K., Grundy, J., 2019. Emotion-oriented requirements engineering: A case study in developing a smart home system for the elderly. J. Syst. Softw. 147, 215–229.

Erdogmus, H., Melnik, G., Jeffries, R., 2010. Test-driven development. In: Encyclopedia of Software Engineering. Taylor & Francis, pp. 1211–1229.

Fagerland, M.W., 2012. t-tests, non-parametric tests, and large studies–A paradox of statistical practice? BMC Med. Res. Methodol. 12 (1), 78.

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., Juristo, N., 2016a. A dissection of the test-driven development process: Does it really matter to test-first or to test-last? IEEE Trans. Softw. Eng. 43 (7), 597–614.

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., Juristo, N., 2017. A dissection of the test-driven development process: Does it really matter to test-first or to test-last? IEEE Trans. Softw. Eng. 43 (7), 597–614.

Fucci, D., Girardi, D., Novielli, N., Quaranta, L., Lanubile, F., 2019. A replication study on code comprehension and expertise using lightweight biometric sensors. In: International Conference on Program Comprehension. IEEE, pp. 311–322.
Fucci, D., Romano, S., Baldassarre, M.T., Caivano, D., Scanniello, G., Turhan, B., Juristo, N., 2018. A longitudinal cohort study on the retainment of test-driven development. In: Proceedings of International Symposium on Empirical Software Engineering and Measurement. ACM, pp. 18:1–18:10.
Fucci, D., Scanniello, G., Romano, S., Shepperd, M., Sigweni, B., Uyaguari, F., Turhan, B., Juristo, N., Oivo, M., 2016b. An external replication on the effects of test-driven development using a multi-site blind analysis approach. In: Proceedings of International Symposium on Empirical Software Engineering and Measurement. ACM, pp. 3:1–3:10.
Girardi, D., Novielli, N., Fucci, D., Lanubile, F., 2020. Recognizing developers' emotions while programming. In: Proceedings of International Conference on Software Engineering. ACM, pp. 666–677.
Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P., 2018. What happens when software developers are (un) happy. J. Syst. Softw. 140, 32–47.
Graziotin, D., Wang, X., Abrahamsson, P., 2013. Are happy developers more productive? In: Proceedings of International Conference on Product-Focused Software Process Improvement. Springer, pp. 50–64.
Herbon, A., Peter, C., Markert, L., Van Der Meer, E., Voskamp, J., 2005. Emotion studies in HCI-a new approach. In: Proceedings of International Conference on Human-Computer Interaction.
Higgins, J.P.T., Thomas, J., Chandler, J., Cumpston, M., Li, T., Page, M.J., Welch, V.A., 2019. Cochrane Handbook for Systematic Reviews of Interventions. John Wiley & Sons.
Höst, M., Regnell, B., Wohlin, C., 2000. Using students as subjects—A comparative study of students and professionals in lead-time impact assessment. Empir. Softw. Eng. 5 (3), 201–214.
Islam, M.R., Zibran, M.F., 2018. DEVA: Sensing emotions in the valence arousal space in software engineering text. In: Proceedings of Symposium on Applied Computing. ACM, pp. 1536–1543.
Jeffries, R., Melnik, G., 2007. Guest editors' introduction: TDD–The art of fearless programming. IEEE Softw. 24 (3), 24–30.
Judd, C.M., 2001. Moderator variable: Methodology. In: International Encyclopedia of the Social & Behavioral Sciences. Pergamon, pp. 9937–9939.
Kaptein, M.C., Nass, C., Markopoulos, P., 2010. Powerful and consistent analysis of likert-type ratingscales. In: Proceedings of International Conference on Human Factors in Computing Systems. ACM, pp. 2391–2394.
Karac, I., Turhan, B., 2018. What do we (really) know about test-driven development? IEEE Softw. 35 (4), 81–85.
Kintis, M., Papadakis, M., Papadopoulos, A., Valvis, E., Malevris, N., Le Traon, Y., 2018. How effective are mutation testing tools? An empirical analysis of Java mutation testing tools with manual analysis and real faults. Empir. Softw. Eng. 23 (4), 2426–2463.
Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., Pohthong, A., 2017. Robust statistical methods for empirical software engineering. Empir. Softw. Eng. 22 (2), 579–630.
Koelstra, S., Muhl, C., Soleymani, M., Lee, J.-S., Yazdani, A., Ebrahimi, T., Pun, T., Nijholt, A., Patras, I., 2012. DEAP: A database for emotion analysis using physiological signals. IEEE Trans. Affect. Comput. 3 (1), 18–31.
Lumley, T., Diehr, P., Emerson, S., Chen, L., 2002. The importance of the normality assumption in large public health data sets. Annu. Rev. Public Health 23, 151–169.
Madeyski, L., Kitchenham, B., 2018. Effect sizes and their variance for AB/BA crossover design studies. Empir. Softw. Eng. 23 (4), 1982–2017.
Mäntylä, M., Adams, B., Destefanis, G., Graziotin, D., Ortu, M., 2016. Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity? In: Proceedings of International Conference on Mining Software Repositories. ACM, pp. 247–258.
McCulloch, C.E., Neuhaus, J.M., 2011. Misspecifying the shape of a random effects distribution: Why getting it wrong may not matter. Stat. Sci. 26 (3), 388–402.
Morris, J.D., Woo, C., Geason, J.A., Kim, J., 2002. The power of affect: Predicting intention. J. Advert. Res. 42 (3), 7–17.
Müller, S.C., Fritz, T., 2015. Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress. In: Proceedings of International Conference on Software Engineering, vol. 1, pp. 688–699.
Paige, R.F., Cabot, J., Ernst, N.A., 2017. Foreword to the special section on negative results in software engineering. Empir. Softw. Eng. 22 (5), 2453–2456.
Romano, S., Fucci, D., Baldassarre, M.T., Caivano, D., Scanniello, G., 2019. An empirical assessment on affective reactions of novice developers when applying test-driven development. In: Proceedings of International Conference on Product-Focused Software Process Improvement. Springer, pp. 3–19.
Romano, S., Fucci, D., Scanniello, G., Turhan, B., Juristo, N., 2016. Results from an ethnographically-informed study in the context of test driven development. In: Proceedings of International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 10:1–10:10.
Romano, S., Fucci, D., Scanniello, G., Turhan, B., Juristo, N., 2017. Findings from a multi-method study on test-driven development. Inf. Softw. Technol. 89, 64–77.
Romano, S., Scanniello, G., Baldassarre, M.T., Fucci, D., Caivano, D., 2020. Results from a replicated experiment on the affective reactions of novice developers when applying test-driven development. In: International Conference on Agile Software Development. Springer, pp. 223–239.
Russell, J.A., Mehrabian, A., 1977. Evidence for a three-factor theory of emotions. J. Res. Personal. 11 (3), 273–294.
Santos, A., Vegas, S., Dieste, O., Uyaguari, F., Tosun, A.e, Fucci, D., Turhan, B., Scanniello, G., Romano, S., Karac, I., Kuhrmann, M., Mandić, V., Rama, R., Pfahl, D., Engblom, C., Kyykka, J., Rungi, K., Palomeque, C., Spisak, J., Oivo, M., Juristo, N., 2021. A family of experiments on test-driven development. Empir. Softw. Eng. 26 (3), 42.
Santos, A., Vegas, S., Oivo, M., Juristo, N., 2019. A procedure and guidelines for analyzing groups of software engineering replications. IEEE Trans. Softw. Eng. 1.
Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J.A., Tortora, G., Risi, M., Dodero, G., 2018. Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments. Empir. Softw. Eng. 23 (5), 2695–2733.
Shull, F.J., Carver, J.C., Vegas, S., Juristo, N., 2008. The role of replications in empirical software engineering. Empir. Softw. Eng. 13 (2), 211–218.
Stade, M., Scherr, S.A., Mennig, P., Elberzhager, F., Seyff, N., 2019. Don't worry, be happy – Exploring users' emotions during app usage for requirements engineering. In: Proceedings of International Requirements Engineering Conference. IEEE, pp. 375–380.
Thurmond, V., 2001. The point of triangulation. J. Nurs. Scholarsh..
Turhan, B., Layman, L., Diep, M., Erdogmus, H., Shull, F., 2010. How effective is test-driven development. In: Making Software: What Really Works, and Why We Believe It. O'Reilly Media, pp. 207–217.
Vegas, S., Apa, C., Juristo, N., 2016. Crossover designs in software engineering experiments: Benefits and perils. IEEE Trans. Softw. Eng. 42 (2), 120–135.
Wellek, S., Blettner, M., 2012. On the proper use of the crossover design in clinical trials. Dtsch. Arztebl. Int. 109 (15), 276–281.
Whitehead, A., 2003. Meta-Analysis of Controlled Clinical Trials. John Wiley & Sons.
Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A., 2012. Experimentation in Software Engineering. Springer.

Maria Teresa Baldassarre is Associate Professor at the University of Bari, Italy, and member of the Software Engineering Research Laboratory (SERLab). Her research interests are: empirical software engineering, human factors in software engineering, quality assessment, improvement and management in software processes, products and projects. She collaborates on several research projects and carries out controlled and in field experimentation within small and medium enterprises. She is a partner of the SER&Practices spin off company of the University of Bari. She is actively involved in research projects and collaborations with international partners. Currently she is the representative of the University of Bari in the International Software Engineering Research Network (ISERN), and is involved in various program committees related to relevant software engineering and international empirical software engineering venues.

Danilo Caivano is Professor of Software Engineering and Project Management at the Department of Computer Science of the University of Bari Aldo Moro, as well as consultant for companies and organizations mainly in the context of research and development projects. He is currently head of the SERLAB research laboratory, director of the short master in cyber security, he has contributed to the creation of The Hack Space, the cyber security laboratory of the University of Bari. He is a member of the Board of Director of the Southern Italy Chapter Project Management Institute and coordinator of the PMI-SIC Academy. He is a member of the Technical Scientific Committee of the Apulian Information Technology District and of the IT Strategic Steering Committee. He is the representative of the University of Bari in the International Software Engineering Research Network (ISERN).

Davide Fucci is an Assistant Professor at Blekinge Institute of Technology (Sweden). He received his Ph.D. from the University of Oulu (Finland) in 2016. His research interests lie in data-drive requirements engineering, test automation, security testing, and human aspects of software development. Dr. Fucci has published over 50 articles in international journals and conferences and received several best paper awards. He has served on the program committees of over 20 academic conferences, on the editorial or review boards of several top-tier software engineering journals. He started the AffectRE workshop series on emotional awareness in requirements engineering. He is a member of ACM, ACM SIGSOFT, IEEE and IEEE Computer Society. For more information please visit: http://dfucci.co.

Simone Romano received his Ph.D. in Computer Science and Mathematics from the University of Salento, Italy—in collaboration with the University of Basilicata, Italy—in July 2018. After a few months, he joined the Department of Computer Science at the University of Bari, Italy, as a postdoctoral research fellow. Since December 2020, he has been a (fixed-term) assistant professor at the University of Bari. He has served in the organization and has been a program committee member of several international conferences/workshops in the Software Engineering research field. He has also been a reviewer for international journals in that research field.

Giuseppe Scanniello received his Laurea and Ph.D. degrees, both in Computer Science, from the University of Salerno, Italy, in 2001 and 2003, respectively. In 2006, he joined, as an Assistant Professor, the Department of Mathematics and Computer Science at the University of Basilicata, Potenza, Italy. In 2015, he became an Associate Professor at the same university. He has published more than 170 referred papers in journals, books, and conference proceedings. He serves on the organizing of major international conferences and workshops in the field of software engineering. Giuseppe Scanniello leads both the group and the laboratory of software engineering at the University of Basilicata (BASELab).