RESEARCH-ARTICLE

Relevant Information in TDD Experiment Reporting

FERNANDO UYAGUARI

SILVIA T. ACUÑA, Autonomous University of Madrid, Madrid, Madrid, Spain

JOHN W. CASTRO, University of Atacama, Copiapo, AT, Chile

DAVIDE FUCCI, Blekinge Institute of Technology, Karlskrona, Blekinge, Sweden

OSCAR DIESTE, Technical University of Madrid, Madrid, Madrid, Spain

SIRA VEGAS, Technical University of Madrid, Madrid, Madrid, Spain

Open Access Support provided by:

Blekinge Institute of Technology

Autonomous University of Madrid

Technical University of Madrid

University of Atacama

Published: 25 January 2025
Online AM: 22 August 2024
Accepted: 12 July 2024
Revised: 02 June 2024
Received: 05 December 2023

Relevant Information in TDD Experiment Reporting

FERNANDO UYAGUARI, Instituto Superior Tecnológico Wissen, Cuenca, Ecuador
SILVIA T. ACUÑA, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain
JOHN W. CASTRO, Departamento de Ingeniería Informática y Ciencias de la Computación, Universidad de Atacama, Copiapó, Chile
DAVIDE FUCCI, Blekinge Institute of Technology, Blekinge, Sweden
OSCAR DIESTE and SIRA VEGAS, Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Madrid, Spain

Experiments are a commonly used method of research in software engineering (SE). Researchers report their experiments following detailed guidelines. However, researchers do not, in the field of test-driven development (TDD) at least, specify how they operationalized the response variables and, particularly, the measurement process. This article has three aims: (i) identify the response variable operationalization components in TDD experiments that study external quality; (ii) study their influence on the experimental results; (iii) determine if the experiment reports describe the measurement process components that have an impact on the results. We used two-part sequential mixed methods research. The first part of the research adopts a quantitative approach applying a statistical analysis of the impact of the operationalization components on the experimental results. The second part follows with a qualitative approach applying a systematic mapping study (SMS). The test suites, intervention types and measurers have an influence on the measurements and results of the statistical analysis of TDD experiments in SE. The test suites have a major impact on both the measurements and the results of the experiments. The intervention type has less impact on the results than on the measurements. While the measurers have an impact on the measurements, this is not transferred to the experimental results. On the other hand, the results of our SMS confirm that TDD experiments do not usually report either the test suites, the test case generation method, or the details of how external quality was measured. A measurement protocol should be used to ensure that the measurements made by different measurers are similar. It is necessary to report the test cases, the experimental task and the intervention type in order to be able to reproduce the measurements and statistical analyses, as well as to replicate experiments and build dependable families of experiments.

CCS Concepts: • **General and reference** → **Measurement**; **Metrics**; **Experimentation**; **Empirical studies**;

This research was funded by grant PID2022-137846NB-I00 funded by MCIN/AEI/10.13039/501100011033, by "ERDF A way of making Europe" and the FINESSE project, Spain (PID2021-122270OB- I00). This research was also supported by the Madrid Region R & D program, Spain (project FORTE, P2018/TCS-4314) and the SATORI-UAM project (TED2021-129381B-C21).
Authors' Contact Information: Fernando Uyaguari, Instituto Superior Tecnológico Wissen, Cuenca, Ecuador; e-mail: fernando.uyaguari@wissen.edu.ec; Silvia T. Acuña, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain; e-mail: silvia.acunna@uam.es; John W. Castro (corresponding author), Departamento de Ingeniería Informática y Ciencias de la Computación, Universidad de Atacama, Copiapó, Chile; e-mail: john.castro@uda.cl; Davide Fucci, Blekinge Institute of Technology, Blekinge, Sweden; e-mail: davide.fucci@bth.se; Oscar Dieste, Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Boadilla del Monte, Madrid, Spain; e-mail: odieste@fi.upm.es; Sira Vegas, Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Boadilla del Monte, Madrid, Spain; e-mail: svegas@fi.upm.es.

ACM 1557-7392/2025/1-ART28
https://doi.org/10.1145/3688837

Additional Key Words and Phrases: experiment, systematic mapping study, SMS, missing information, test-driven development, TDD, operationalization, measurement, test cases, experimental task, code intervention, measurer

ACM Reference format:
Fernando Uyaguari, Silvia T. Acuña, John W. Castro, Davide Fucci, Oscar Dieste, and Sira Vegas. 2025. Relevant Information in TDD Experiment Reporting. *ACM Trans. Softw. Eng. Methodol.* 34, 2, Article 28 (January 2025), 41 pages.
https://doi.org/10.1145/3688837

1 Introduction

In order to be able to test an experimental hypothesis, it has to be operationalized [8]. Operationalization is the process of connecting the conceptual definition of the hypothetical variables related to a specific set of measurement procedures [50]. The procedures (and often the associated measurement instruments) depend on the area addressed. For example, test cases are usually used to measure the quality of code developed using **test-driven development (TDD)**, whereas the quality of a specification could be determined using a code review checklist. The following measurement instruments are used in **software engineering (SE)**: questionnaires, measurement software, expert evaluation, etc. In this study, we focused on the operationalization of the external quality response variable in TDD experiments. This is not fortuitous. We conducted a number of TDD experiments [16, 22, 57, 58] and observed that the use of one or other procedure/instrument affects the external quality measurement [17].

External quality refers to all the software product characteristics from an external viewpoint [32]. According to ISO/IEC 205010:2011 [33], the software product external quality model is composed of characteristics and subcharacteristics. In TDD experiments, external quality refers to a combination of software product quality model completeness and functional correctness.

1.1 Research Motivation

This research was motivated by the fact that TDD experiments studying external quality produce different, and sometimes even contradictory, results. For example, studies by Geras et al. [27] and George and Williams [26] suggest that TDD improves quality. However, Huang and Holcombe [30] and Desai et al. [15] claim that the difference is very small and is not statistically significant. Madeyski [40] states that TDD reduces external quality. Researchers have conducted secondary studies examining the experiment context (industry or academia) [10, 55] or the study's rigour (research method applied according to best practice) and relevance (real impact on industry) [45] in search of explanations for these inconsistencies, although the results are not absolutely satisfactory.

In our opinion, some of the differences in the results of TDD experiments that study external quality could be due to measurement-related issues [17]. The instruments usually used in TDD experiments to measure external quality are typically acceptance test cases [16, 58]. External quality is proportional to the number of acceptance test cases passed [55]. However, many aspects related to the test cases are almost certain to vary across experiments, including, for example, the test case generation technique (**ad hoc (AH)**, **equivalence partitioning (EP)**, etc.), the person applying the test cases to the source code (subject, researcher, etc.), or the problem-solving criteria in the event of errors (method parameter inconsistency, etc.). TDD analyses carried out by the authors of secondary studies (e.g., [4, 45]) have not taken into account these differences. In fact, the test cases used to measure the quality of the code generated by subjects could differ across studies where the measurement process components and the impact of such differences are not well understood and

are, as a result, passed over by secondary studies. The same applies to other differences in external quality measurement.

1.2 Problem-Solving Approach

The aim of our study is to identify the components of the response variable operationalization in TDD experiments that study external quality, investigate their influence on the experimental results, and determine whether the experimental reports describe the measurement process components that have an impact on the results in sufficient detail.

We used two-stage sequential mixed methods research. The first stage adopts a quantitative approach based on statistically analysing the impact of the operationalization components on experimental results. The second stage follows up with a qualitative approach using a **systematic mapping study (SMS)**, which tests and rounds out earlier findings on TDD experiment reporting.

The first part of the research was carried out on the data of the experiment called **FSecure (FS)**, described in [65], which we conducted with other colleagues. The aim of the FS experiment was to compare the TDD and **incremental test last (ITL)** techniques with respect to software external quality and developer productivity in a software company setting from the viewpoint of researchers. Twenty-four professionals participated in the experiment conducted on FS premises in three different cities. As we have access to the data from the FS experiment, we can experimentally analyse whether the test suites, intervention types and measurers had an influence on the TDD measurements.

The second phase of the sequential research was conducted by means of a secondary study. This study examined how TDD experiments operationalized the *external quality* response variable and to what extent the empirical evidence gathered in the FS experiment is applicable to TDD experiments.

1.3 Research Contributions

Our findings are relevant for experimental TDD researchers and may possibly be extrapolated to other areas of SE. The contributions of this article are as follows:

- We have demonstrated that the test cases and intervention type used to measure the value of the external quality variable in TDD experiments influence the experimental results, that is, the statistical analyses yield different conclusions depending on the test case or intervention type used. Although these observations were made on TDD experiments, there is no reason to suspect that these effects do not also occur in other SE experiments.
- We have proposed improvements for the measurement of TDD experiments, which could (probably) be extrapolated to SE as a whole: (1) whenever possible, use different components, e.g., test cases created by different researchers using different strategies; (2) carry out repeatability and reproducibility studies; (3) check that the response variable satisfies the *representation condition* [21, Chapter 2]; (4) automate measurements as far as possible; (5) apply blinding techniques to prevent bias.
- We suggest improvements for experimental reporting in TDD, which, again, could be applicable to other SE experiments. At the very least, the test cases, experimental task and intervention type need to be reported to be able to reproduce the measurements and analysis. The measurement process should be described in detail.

This paper is organized as follows. Section 2 describes the background. Section 3 describes the experimental set-up and SE measurement process used in this paper. Section 4 defines the FS experimental measurement process. Section 5 describes the research objectives, the mixed methods research process, and the quantitative and qualitative research questions. Section 6 evaluates the impact of the measurement process components on the experimental results in TDD. Section 7

reports the result of the SMS. Section 8 defines the validity threats. Section 9 discusses the results. Finally, Section 10 describes the conclusions and future work.

2 Background

A hypothesis on the cause-effect relationship represents the belief that there is a relationship between a cause construct and an effect construct [68]. Scientific constructs often refer to imperceptible events [8]. In order to test a hypothesis, it has to be operationalized. Operationalization is the process of connecting a conceptual definition with a specific set of measurement techniques or procedures [50]. For example, the operationalization of the hypothesis:

TDD improves external quality

implies:

—Deciding what TDD means, that is, test-first vs. test-first+refactoring, which will determine the treatment implemented in the experiment.
—Selecting a response variable that represents the "external quality" construct.
—Establishing a response variable measurement process.

Researchers usually use more specific processes than the abstract concept of operationalization. The measurement process has been addressed at length in SE, which is attested to by Fenton and Bieman's well-known book [21], along with many other publications that we will not cite here. However, the impact of the measurement process on practice has not received very much attention. This contrasts with other disciplines, such as medicine, where studies on the impact of measurers on measurement results (e.g., [28, 62]) are readily available. The Bland-Altman plot [5] used in Section 6 was originally developed to compare measurement instruments in medicine. There are many others.

Focusing on SE, warnings about the impact of measurement on experimentation came fairly early on. For example, *"The results of an experiment can be no more valid than the measurement of the constructs investigated. [...] Many studies have elaborately defined the independent variables (e.g., the software practice to be varied) and hastily employed a handy but poorly developed dependent measure (criterion). Results from such experiments, whether significant or not, are difficult to explain"* [14]. Kitchenham et al. [37] stated that it was necessary to critically review and evaluate all the research in this field.

In the following years, however, measurement problems received less attention. One possible reason, according to Abran et al. [1], is that measurement in SE has been carried out mainly from the viewpoint of mathematical measurement theory. This has meant that key measurement aspects, like measurement method and measurement instrument, have remained in the background. They defend the adoption by SE [34] of practices proper to metrology, which do account for the above aspects.

With respect to the definition of *metric*, the terms *measurement* and *metric* are often used interchangeably in SE. The concepts related to measurement are defined in the **Vocabulaire International de Métrologie (VIM)**.[1] This standard defines *measurement* as the process of experimentally obtaining one or more quantity values from a measurand. VIM does not include the term *metric*, because this is a mathematical concept that expresses the distance between two points [21, p. 120].

Some SE authors have shown concern for the practical problems of measurement. Kitchenham et al. [38] signal that, apart from relying on formally well-defined variables, the measurement

[1] https://www.bipm.org/documents/20126/2071204/JCGM_200_2012.pdf/f0e1ad45-d337-bbeb-53a6-15fe649d0ff1

process needs to be underpinned by a measurement protocol that *"defines the who, when, and how of any data collection activity, i.e., (1) Who: the role responsible for data extraction; (2) When: the point in the development process when the measure should be taken; (3) How: what tools, methods are used to extract, record and store the data values"*. (We substitute the bullet items for numbers in the citation.) Kitchenham et al. [39] reiterated, perhaps less clearly, the above recommendations in a later, well-known article.

In this study, we focus on the operationalization of the above-mentioned external quality response variable. Different researchers may choose different response variables to represent *external quality* (e.g., test cases run, number of satisfied requirements) and different procedures for measuring even the same response variable (e.g., manual testing vs. automated testing). The experimental result should be the same irrespective of how it is measured. If the procedure affects the experimental results, the results are not valid [68]. This raises the question of *whether decisions made by researchers could have an impact on experimental results*. Similar arguments could likewise apply to the operationalization of the TDD treatment and have, in fact, already been addressed as part of the study of TDD experiment conformance [24].

3 Experimental Set-Up

In order to answer the above question, we use the FS experiment as a case study. We ran the FS experiment described in [65], together with other colleagues. The aim of the FS experiment was to compare the TDD and ITL techniques with respect to external quality and developer productivity.

3.1 Variables

Two response variables were used in the FS experiment: external quality and productivity. In this study, we focus on external quality, as it is widely used in TDD experiments. However, the results are equally applicable to productivity. In the FS experiment, we used the same variables and measurements as in [19, 23], where external quality was defined as a combination of software product quality model completeness and functional correctness as defined in ISO/IEC 25010:2011 [33].

It is also necessary to specify which direct measurements are necessary and which measurements can be derived from the direct measurements. The external quality measurement is calculated using the tackled subtasks (tst) of a given task.[2] A subtask is considered to have been tackled if it passes at least one assertion in the test case suite associated with the subtask. This measurement unit can differentiate between the subtasks that the experimental subject has and has not tried to complete. tst is calculated using Equation (1):

$$tst_i = \begin{cases} 1 & \text{if } \#Assert_i(PASS) > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where n, is the total number of subtasks that make up the experimental task. The quality of the ith subtask is calculated using Equation (2) below:

$$QLTY_i = \frac{\#Assert_i(PASS)}{\#Assert_i(ALL)}. \tag{2}$$

$\#Assert_i(PASS)$ represents the total number of assertions that pass the acceptance test cases associated with the ith subtask. $\#Assert_i(ALL)$ symbolizes the total number of assertions created to test the ith subtask.

Intuitively, the quality of the experimental task should be the sum of the quality of its subtasks (probably divided by $\sum_{i=1}^{n} tst_i$, i.e., the number of tackled subtasks). However, this would mean that

[2] For a definition of the experimental tasks, see https://github.com/GRISE-UPM/TOSEM-TestSuitesMeasurement/blob/master/experimental_tasks

the simpler subtasks with fewer associated assertions would carry a greater weight than the more complex subtasks with more assertions with respect to the final quality. To avoid this issue, we calculate the final quality as indicated in Equation (3) below:

$$QLTY = \frac{\sum_{tst_i=1} \#Assert_i(PASS)}{\sum_{tst_i=1} \#Assert_i(ALL)}. \tag{3}$$

3.2 Experimental Design and Tasks

A pre-test/post-test design was used. All the subjects applied both treatments in a predetermined order. ITL and TDD were applied on the **MarsRover API (MR)** and **bowling score keeper (BSK)** experimental tasks, respectively. Both MR and BSK are popular exercises used by the agile community.

3.3 Instrumentation

Java was used as the programming language and JUnit 4, as the testing framework. The subjects received the task specification and a source code template with a class structure and common methods to facilitate the measurement process. Measurement was made using acceptance test suites, and the measurement unit was the number of assertions passed.

Although we take acceptance test suites for granted, the fact is that, with the exception of experimental replications, different experiments use different test cases. *This is the primary reason for this research.* Test cases are not standard; they are designed by researchers. The use of different test cases can lead to sizeable measurement differences [17].

The FS experiment test cases are available at GitHub.[3] One test suite was reused from previous experiments [19, 65]. It was generated using an AH strategy and coded in JUnit. AH means that no formal procedure was used to create the test cases; the test suite authors (for MR and BSK) used their best judgement to write a set of test cases based on the functional requirements.

3.4 Measurement Example

Below, there follows a measurement example based on the BSK experimental task, which is used to run the above equations. We selected subtasks 1 and 2 for this example:

```
1. Frame : Each turn of a bowling game is called a frame. 10 pins are
arranged in each frame. The goal of the player is to knock down as
many pins as possible in each frame. The player has two chances, or
throws, to do so. The value of a throw is given by the number of pins
knocked down in that throw. Requirement: Define a frame as composed
of two throws. The first and second throws should be distinguishable.

Example: [2, 4] is a frame with two throws, in which two pins were
knocked down in the first, and four pins were knocked down
in the second throw.

2. Frame Score: An ordinary frame score is the sum of its throws.

Requirement: Compute the score of an ordinary frame.

Examples: The score of the frame [2, 6] is 8. The score of the frame
[0, 9] is 9.
```

[3]The test suites are provided as a single Eclipse workspace containing four projects. They are available as one Eclipse workspace at https://github.com/GRISE-UPM/TestSuitesMeasurement/tree/master/test_suites

Table 1. Characteristics of AH and EP Test Suites

Task		Test suite AH	EP
MR	Test classes	11	9
	Test methods	52	32
	Assertions	89	32
BSK	Test classes	13	13
	Test methods	48	72
	Assertions	55	72

```
package upm.tdd.training;

public class US01 {
  @Test
  public void testEmptyFrameIsCreated () {
    Frame f = new Frame();
    assertNotNull (f) ;
  }
  @Test
  public void testFrameWithScoreIsCreated() {
    Frame f = new Frame(1,2);
    assertNotNull(f);
  }
}

public class US02 {
  @Test
  public void testFrameScore1()  {
    Frame f = new Frame(0,0);
    assertEquals(0, f.score());
  }
  @Test
  public void testFrameScore2()  {
    Frame f = new Frame(2,4);
    assertEquals(6, f.score());
  }
  @Test
  public void testAllPinsFrameScore() {
    Frame f = new Frame(10,0);
    assertEquals(10, f.score());
  }
}
```

Fig. 1. Test cases for BSK subtasks 1 and 2. This code was simplified and organized for comprehensibility. Note that the experimental subjects do not have access to this test code.

The test suite associated with the AH strategy contains 89 assertions, divided into 13 test classes (see Table 1) Each test class matches one of the subtasks of the BSK experimental task. The test code corresponding to subtasks 1 and 2 is shown in Figure 1.

Supposing, for a moment, that we were to solve BSK subtask 1. We might well have written code resembling that shown in Figure 2: we would have just a `Frame` class with a `Frame(int i, int j)`

```
package upm.tdd.training;

public class Frame {
  public Frame(int i, int j) {
    // TODO Auto-generated constructor stub
  }
}
```

Fig. 2. First `Frame` class coding step using TDD.

constructor (automatically generated in this case by the *Eclipse* IDE). When the US01 class tests (associated with subtask 1) are run, we observe that the `testEmptyFrameIsCreated ()` method assertion fails, but the `testFrameWithScoreIsCreated()` method assertion passes. All the `US02` class assertions fail. According to Equation (1), the number of tackled subtasks is calculated as the number of subtasks for which at least one associated test case assertion passes. Therefore, as all the assertions of subtasks 3-13 fail, the values of tst_i are:

$$tst_i = \begin{cases} 1 & \text{if } \#Assert_i(PASS) > 0 \\ 0 & \text{otherwise} \end{cases} = \begin{cases} tst_1 = 1 \\ tst_2 = 0 \\ tst_i = 0 & \text{if } 2 < i \leq 13 \end{cases}$$

Intuitively, if none of the assertions associated with subtask 2 pass, the quality of this subtask should be zero. With respect to subtask 1, the quality should be intermediate. According to Equation (2), the values of $QLTY_i$ are:

$$QLTY_1 = \frac{\#Assert_1(PASS)}{\#Assert_1(ALL)} = \frac{1}{2}$$
$$QLTY_2 = \frac{\#Assert_2(PASS)}{\#Assert_2(ALL)} = \frac{0}{3}$$

By definition, $QLTY_i$ (like $QLTY$) has a range $[0 \ldots 1]$. In practice, we express $QLTY_i$ values as percentages, as it is more intuitive. Accordingly, $QLTY_1 = 50\%$ and $QLTY_2 = 0\%$. External quality is calculated using Equation (3). This equation implies summing the number of assertions passed in the tackled subtasks divided by the total number of assertions of the tacked subtasks. As only $QLTY_1$ has been "tackled," that is, $tst_1 = 1$, the total number of assertions to consider is two, of which one has passed:

$$QLTY = \frac{\sum_1 \#Assert_i(PASS)}{\sum_1 \#Assert_i(ALL)} = \frac{1}{2} = 0.5, \text{ or } 50\%$$

When the programmer solves the subtasks, the external quality increases. For example, Figure 3 shows the code of the `Frame` class resulting from using TDD to complete a couple of `score()` method coding steps (*fake code using constants and generalize later* [2, p. 13]). Such a code implies that the `testFrameScore1()` method assertion passes. Therefore, the status of subtask 2 is now "tackled", that is, $tst_2 = 1$, and external quality would be:

$$QLTY = \frac{\sum_{1,2} \#Assert_i(PASS)}{\sum_{1,2} \#Assert_i(ALL)} = \frac{1+1}{2+3} = \frac{2}{5} = 0.4, \text{ or } 40\%$$

Finally, note that the `testEmptyFrameIsCreated()` method assertion will never pass, as there is no reason to create a `Frame()` constructor in the `Frame` class. This test case limits the external quality range. The maximum quality value using classes US01 and US02 ranges between

```
package upm.tdd.training;

public class Frame {
  public Frame(int i, int j) {
    // TODO Auto-generated constructor stub
  }
  //the score of a single frame
  public int score(){
    return 0;
  }
}
```

Fig. 3. Second `score()` method coding step using TDD.

$^{0+0}/_{2+3}$ = 0% and $^{1+3}/_{2+3}$ = 80%. In other words, the `testEmptyFrameIsCreated()` method posed a problem for BSK experimental task quality measurement.

4 FS Experiment Measurement Process

Each of the measurement process activities was divided into several steps, which are shown on the left-hand side of Figure 4. They are mapped to the steps of the measurement procedure enacted in the FS experiment, illustrated on the right-hand side of Figure 4. This clearly depicts how many details have to be taken into account in a real measurement process.

According to Fenton and Bieman [21], the measurement process is composed of four activities: (i) Planning for data collection, (ii) Data collection, (iii) Extraction, and (iv) Analysis. The steps of each of these activities are highlighted (on the left-hand side of Figure 4) in blue, orange, violet and green, respectively. The measurement procedure that we applied in the FS experiment is composed of two steps: (i) Planning for data collection (Select the metric, Prepare the measurement instrument) and (ii) Data collection, extraction and analysis (Measure). The steps of each of these activities are highlighted on the right-hand side of Figure 4) in the same colour as used in Fenton and Bieman's measurement process.

4.1 Planning for Data Collection

Decide which attributes to measure: This step is equivalent to the operationalization that we described in Section 3 and corresponds to *Devise a response variable* and *Define the metric* on the basis of which the experimental tasks are designed and the measurement instruments implemented. For the sake of precision, we use the terms *measurement* and *measurement unit*.

Control configuration: We should make sure that we know everything there is to know about the object to be measured. For example, it is common practice to use cross-over designs in TDD experiments. One property of these designs is that each experimental task is performed twice, once using TDD and again using the control treatment. Once the code has been stored, it is far from easy to determine whether or not the experimental task was resolved using TDD. We numbered treatments in order of execution. The code delivered by the participants in the FS experiment is available at GitHub.[4]

Devise a scheme for identifying each entity that is part of the measurement process: It is important to clarify how the products, versions, installations, failures, etc., will be denoted on the data collection

[4]https://github.com/GRISE-UPM/FiDiPro_ESEIL_TDD/tree/master/COMPANY_05

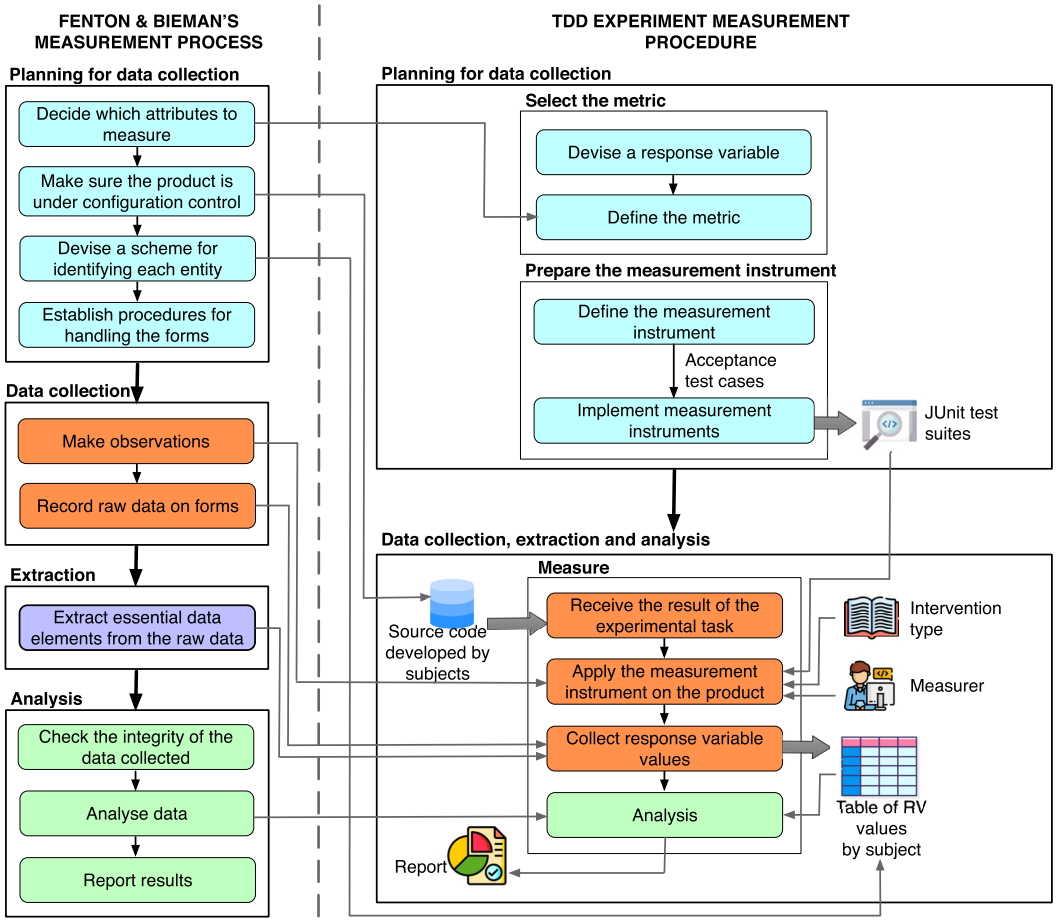


Fig. 4. Measurement process according to Fenton and Bieman [21] vs. our measurement procedure.

forms. In the FS experiment, each participant subject was given a code to assure anonymity. Each column is labelled to identify the measured object and the measurement circumstances. In our case, the columns are source code identifier, experimental task, #tst, quality measurement, session and treatment.

Procedures to handle the forms, analyse the data and report results: It has to be specified who fills in what, when and where, providing a clear description of how the completed forms will be processed. This step is secondary for our purposes.

4.2 Data Collection

Make observations: This step aims to obtain raw data. Raw data are the result of the preliminary measurement of the process, product or resource. The ideal thing is to capture data automatically. In many cases, however, there is no option but to collect data manually. Raw data are the result of applying the acceptance test suites to the code delivered by the experimental subjects. However, the apparent simplicity of this step conceals the other two fundamental problems motivating this research:

(1) *How are conflicts resolved?* The test cases will most probably generate conflicts when applied to the code delivered by the experimental subjects. Common problems are method parameter addition and parameter type changes. However, there may be others, ranging from compilation errors to code incompatibility with test cases. We used three strategies in the FS experiment to connect acceptance test cases with the code delivered by subjects:
   —No intervention: The code developed by subjects is measured as delivered, that is, the code is untouched.
   —Syntactic intervention: The measurer makes changes to the code delivered by subjects to prevent minor errors during measurement.
   —Semantic intervention: The measurer interprets the solution coded by the subject and makes major changes to the delivered source code.
(2) *Who resolves the conflicts?* Conflict resolution cannot generally be performed automatically. A measurer must resolve the conflicts, and this may affect measurement.

In other words, we believe that there is a profound gap between the operationalization and measurement of the variable, which SE experimentation has overlooked until now. The measurement instruments used are odd, at least in TDD and possibly in other SE areas. Instead of measurement rules or instruments related to well-defined units of measurement, like forces or electric charges, SE uses units of measurement, like acceptance tests passed, which are unique and whose relationships to the measurement instruments are largely AH. We also believe, as illustrated in Section 6, that these instruments and their application have to be taken into account.

Record raw data on forms: Either automatically (preferably) or manually. In our case, the measurements were stored manually in Excel files, available at Google Sites.[5] The possibility of the measurer entering incorrect data cannot be ruled out, which means that transcription is another source of error that may be amplified by the measurer.

4.3 Extraction

In Step 3, the *essential data elements are extracted from the raw data* so that the analysts can derive attribute values. In the FS experiment, Excel formulas were created to implement Equation (3), and QLTY was measured directly from the raw data. This ruled out errors of calculation.

4.4 Analysis

Finally, it is necessary to *check data integrity and analyse and report the results.*

These are well-known activities regularly performed by the SE community. This also applies to the FS experiment. Besides, the results of these activities are described in detail in [65].

5 Objectives and Methodology

To the best of our knowledge, the impact of the measurement procedure on experimental results has not yet been studied in SE and definitely not in TDD.

The objective of this research is to determine what impact the measurement method components have on the experimental results.

To do this, we must first ascertain which measurement components may affect the measurement results and by how much. Now, if the identified components were not routinely used in TDD experiments, their impact would be low. Therefore, it is necessary to study the TDD literature to find out how many experiments may be affected. It is the measurement components identified initially that determine the type of review to be carried out, assuring that the research as a whole is consistent.

[5] https://sites.google.com/site/fidiproeseil/home

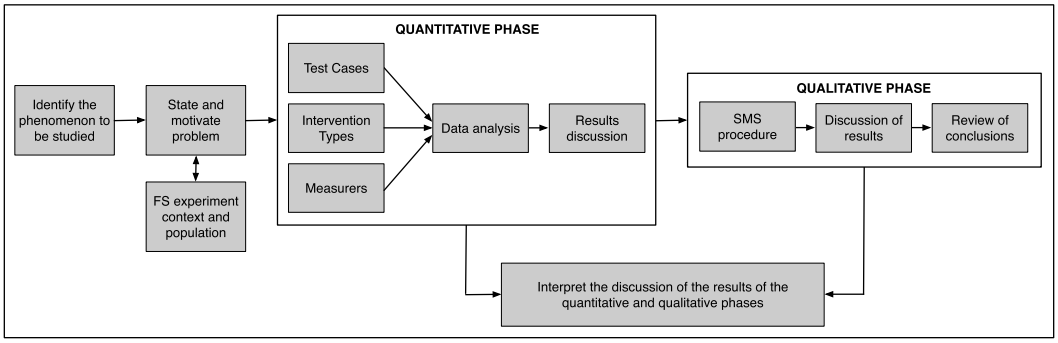


Fig. 5. Sequential mixed methods schema.

Therefore, we designed a sequential mixed methods research approach. This decision was made because the use of different methodological approaches increases the generalizability, certainty and dependability of the conclusions [61, Chapter 29]. Additionally, the use of two methods with their respective strengths and weaknesses that get the same results increases confidence in the results offering a trustworthy representation of the phenomenon under study [64, Chapter 9]. Our design is characterized by a first phase in which quantitative data are collected and analysed, followed by a second stage where qualitative data are collected and evaluated [13].

The first phase adopts a quantitative approach based on statistically analysing the impact of the operationalization components on experimental results. The second phase follows on with a qualitative approach using an SMS.

The method is mixed because the preliminary quantitative results drive the collection of qualitative data. Figure 5 shows an outline of the method that we used.

The quantitative research question is: *(RQ1): What impact do test cases, intervention type and measurers have on the external quality of TDD experiments?* The quantitative procedure includes a series of analyses using the Bland-Altman plot [5] applied to the published dataset of the FS experiment conducted in industry [65] to compare measurements and check the impact of the three components that are overlooked in the TDD external quality measurement process: (i) test case design, (ii) intervention type, and (iii) measurer. This quantitative phase of the mixed design is detailed in Section 6.

The qualitative research question is: *(RQ2): What measurement method components are reported in the experiments on TDD that study external quality?* After identifying the measurement method components that have an impact on response variable measurement, an SMS, conducted following guidelines by Kitchenham and Charters [36], was used to check whether the respective components are reported in the TDD experiments published in the literature. This qualitative phase of the mixed design is detailed in Section 7.

6 Evaluation of the Impact of the Measurement Process on Experimental Results (RQ1)

As mentioned in Section 4, a number of activities have to be performed to enact the measurement process applied to TDD experiments that evaluate external quality. They are:

—Design and code the test suites.
—Connect the code delivered by subject with the test suites, resolving syntactic and semantic inconsistencies.
—Collect the pass/failure information for the test cases.

These activities could influence the results of the measurement. Indeed, Dieste et al. [17] already found that the use of different test cases may have a major impact on the response variable values. As a result, the associated statistical analyses may be significant in one case and not significant in another or the effects may be positive in one case and negative in another. This research, which uses the same methods as [17], confirms the influence of the test cases on measurement and identifies other aspects that influence the *external quality* response variable.

In the following, we quantitatively analyse the influence of the above activities using the dataset of the FS experiment[6] conducted in industry [65]. The main test method that we use is the Bland-Altman plot [5]. Although there are other measurement comparison methods, like ISO 5725 [31], that provide more synthetic results, the Bland-Altman plot is a graphical technique that helps to visualize the differences between measurements. For an overview of the different methods of comparison available, see Dieste et al. [17].

6.1 Test Cases

6.1.1 Description of Test Cases. TDD experiments that study external quality often use test cases [9, 11, 15, 22].[7] The test cases are run on the source code delivered by subjects. The formulas described in Section 6.1.2 are applied to the test cases run to yield the measurement value.

In the FS experiment [65], we relied on an acceptance test case suite for each task (MR and BSK). These test case suites were generated using two different techniques: AH and EP, respectively.[8]

No formal procedure was used to generate the AH test cases. The term *ad hoc* refers to a specific solution developed for a particular problem. The authors of test case suites used their best judgement to create the test case suite.

The EP technique was also used to develop the test case suite. EP is based on the fact that component inputs and outputs can be divided into equivalence classes according to the specifications. There are two possible types of equivalence classes: valid equivalence classes that represent valid inputs for the program and invalid equivalence classes that represent all the other possible states of the condition [47]. The result of testing a single value of the equivalence partition is considered to be representative of the whole partition [7]. Table 1 shows the number of test classes/methods/assertions of each suite for the MR and BSK tasks. All the test case suites were coded in the JUnit test framework.

The result for external quality (QLTY) output by the test suites is a percentage (0%–100%). This percentage represents, for example, the extent to which the code complies with the software requirements: a value of 0% means that the code does not satisfy any requirement, and a value of 100% means that the code meets all the requirements.

Dieste et al. [17] analysed the test case suites and found that AH and EP are more or less equivalent. The statement coverage yielded very similar results (100%) for both AH and EP. Branch coverage also output similar results (90%), except for the AH test case suite applied to the MR task, which yielded slightly lower outcomes (88%). These coverages suggest that the results of testing using both test suites will be similar. For example, a simple correlation analysis to evaluate convergent validity returns large, statistically significant correlations ($r > 0.5$, according to Cohen [12]).

[6]The FS experimental data [65] were completely anonymized. Neither the company that sponsored the experiment, nor the researchers can relate participants to response variable values.

[7]Other studies measure external quality considering the number of defects detected by bug tracking systems [3, 48]. However, these defects may have been identified using a wide range of procedures. Therefore, we focus on test cases.

[8]The test suites are provided as a single Eclipse workspace containing four projects. They are available as one Eclipse workspace at https://github.com/GRISE-UPM/TestSuitesMeasurement/tree/master/test_suites

Table 2. Statistical Analysis of QLTY Response Variable for AH and EP Test Cases

	Ad-hoc	Equivalence partitioning
(Intercept)	63.72 (6.00)***	42.58 (4.50)***
TreatmentTDD-greenfield	21.93 (8.48)*	−12.84 (6.37)*
R^2	.13	.08
Adj. R^2	.11	.06
Num. obs.	48	48

***$p < 0.001$; **$p < 0.01$; *$p < 0.05$.

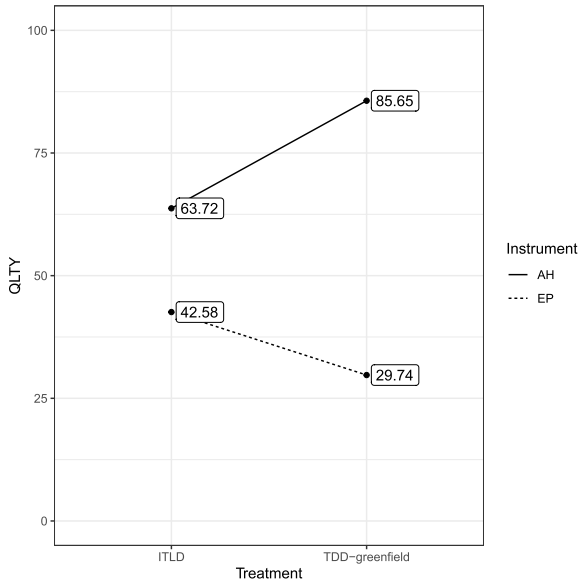


Fig. 6. Effect of ITL/TDD depending on whether the measurement is made using the AH or EP test cases.

6.1.2 Impact on Statistical Analyses. The FS experiment was designed as a pre-test/post-test experiment [60]. The applicable data analysis for this type of experiment is

$$QLTY = Treatment + \epsilon. \tag{4}$$

Table 2 shows the analysis of external quality measured using the AH test suites (originally reported in [65]) and EP. There are clear differences. The intercept is similar (which is to be expected for positive and bounded response variables). However, the effect of TDD is positive for the AH suite (21.93 points) and negative for EP (−12.48 points), and the results are statistically significant in both cases. The profile plot in Figure 6 illustrates the information reported in Table 2 more clearly. The solid and dashed lines represent the effect of TDD as measured by AH and EP, respectively. It is clear that AH and EP yield opposite results, especially for TDD.

6.1.3 Source of the Differences in Statistical Analyses. There is only one possible cause behind the differences in the results of the statistical analyses: there are differences in the values of the *QLTY* response variable for AH and EP. The Bland-Altman plot is a mechanism for visualizing such differences [5].

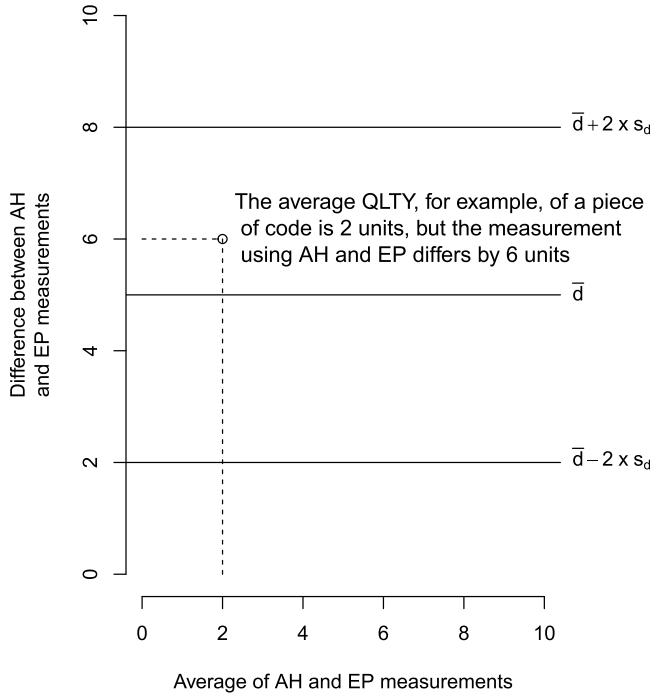


Fig. 7. Difference between AH and EP measurements.

The Bland-Altman method [5] is the de facto standard for comparing measurement instruments in medicine, but it can be easily adopted in SE. In our research problem, QLTY has been measured twice using different instruments: the AH and EP test suites. We thus have two sets of measures $Y_{AH} = \{y_{AH_1}, y_{AH_3}, \ldots, y_{AH_n}\}$ and $Y_{EP} = \{y_{EP_1}, y_{EP_3}, \ldots, y_{EP_n}\}$, where $1 \leq i \leq n$ are the different pieces of code measured. The Bland-Altman method starts with the calculation of the difference between measurements:

$$d_i = (y_{AH_i} - y_{EP_i}). \tag{5}$$

Next, the average of the differences d_i is calculated as:

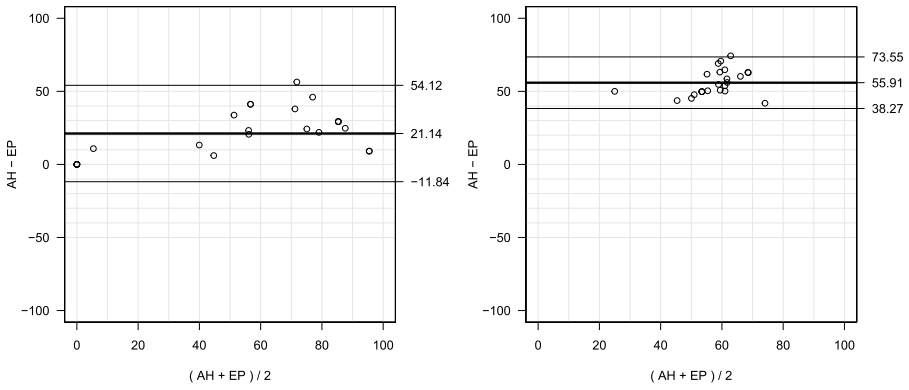
$$\bar{d} = \frac{1}{n} \sum_{i=1}^{n} (y_{AH_i} - y_{EP_i}), \tag{6}$$

$\bar{d}$ represents the mean difference between the measurements obtained with the AH and EP test suites. Finally, the standard deviation is calculated as:

$$s_d = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (d_i - \bar{d})^2}. \tag{7}$$

The Bland-Altman method has an associated graphical representation (the Bland-Altman plot), shown in Figure 7. This graph plots the mean values obtained by both measurement instruments:

$$\frac{y_{AH_i} + y_{EP_i}}{2}, \tag{8}$$



(a) MR (b) BSK

Fig. 8. Bland-Altman plots for differences between AH and EP test cases.

that is, the best estimation of the true measurements against their difference:

$$y_{AH_i} - y_{EP_i}. \tag{9}$$

A horizontal line is drawn at the mean difference $\bar{d}$. Additionally, the graph also depicts two additional horizontal lines located at

$$\bar{d} \pm 2 \times s_d. \tag{10}$$

Assuming a normal distribution for the differences, these bounds enclose 95% of the differences d_i. These bounds represent the variability in the measurement of the same code when one (AH) or other (EP) text suite is used.

The Bland-Altman plots shown in Figure 8 illustrate the distribution of the differences between the measurements made using the AH and EP test cases. In the FS experiment, MR was used as the control task in the first phase of the experiment (ITL trial), and BSK was employed in the second phase (during TDD use). As the treatment and task are confounded, separate Bland-Altman plots need to be created for the two tasks in order to analyse the differences highlighted in Table 2. Figure 8 brings to light the following characteristics:

- —The differences (AH-EP) are generally greater than 0. The measurement values with AH are greater than the measurements using EP.
- —On average, the measurements made with AH are 21.14 points higher than measurements using EP for the MR task. In the case of the BSK task, the differences are much larger, with a difference of 59.91 points.
- —The measurements fluctuate more for MR than for BSK. Figure 8(a) shows that the measurements can vary by ±32.98 points with respect to the average differences. For BSK (see Figure 8(b)), the differences are much smaller (±17.64 points).

There is no simple explanation for why the AH and EP test suites provide such different measurements. For a clarification of the causes, see Dieste at al. [17]. This research is concerned not with the reasons for the above differences but with the fact that they exist and what impact they have on the experimental results, as discussed below.

6.1.4 Impact of the Differences between Measurements on Statistical Analyses. The differences in the measurements made using the AH and EP test cases are a possible explanation for the deviations in the statistical analyses mentioned above:

—The average differences between measurements explain the change in the treatment effect in Table 2. Note that AH produces higher measurements than EP, and these differences are greater for the BSK task. As TDD and BSK are confounded, AH is favourable to TDD, which scores $55.91 - 21.14 = 34.77$ points more than ITL/MR. As Table 2 shows, $-12.84 + 34.77 = 21.93$.
—As measurements made with AH are more variable, its standard deviation is greater than for EP. Although it is not as easy to numerically relate Table 2 and Figure 8(a), in this case, we can see that the standard deviation of the AH measurement (8.48) is greater than for EP (6.37). In this case, the larger variance does not have an effect on the analysis, but it could possibly transform a statistically significant into a statistically non-significant result.

6.2 Intervention Types

6.2.1 Description of Intervention Types. The test cases and the code to be measured can be connected in different ways, which we refer to as *intervention types*. Although there are many possible variants, our experience from the ESEIL project [35] suggests that there are three possible ways of manipulating code for measurement using test cases: (i) no intervention, (ii) syntactic intervention, and (iii) semantic intervention. We describe each intervention type below.

No intervention: In this type of measurement, the code delivered by the subjects is measured as is without any intervention at all by the measurer. Any minor errors in the source code (for example, a missing semicolon) are not corrected.

Syntactic intervention: The measurer makes some changes to fix minor bugs in the source code delivered by the subjects. These errors are easily identifiable in a debugging session, and the measurement values for code containing uncorrected errors would be lower than warranted, as the code containing the minor bug would fail the test cases. The main operations for carrying out a syntactic intervention are as follows:

—Rename the classes of the source code delivered by subjects using the test case labels.
—Rename the methods of the code delivered by subjects using test case labels.
—Correctly assign the variables that return the results of a method.
—Modify the output format of the methods to what is expected by the test cases.
—Create the constructors with the parameters used in the test cases.
—Create the methods used in the test cases in the code delivered by the subjects.
—Assign suggested initial values to the variables, for example: *xStart=0*.

Figure 9 shows an example of source code delivered by an experimental subject. The code represents two functions that return a value (*public int getfirstThrowValue()* and *public int getSecondThrowValue()*). However, these function names are incorrect insofar as they are not the function names used in the measurement test cases. The subject does not appear to have followed the suggested source code template.

Figure 10 shows the required syntactic intervention on the source code: the measurer changes the name of the functions to the names used in the measurement test cases (*public int getThrow1()* and *public int getThrow2()*). Without this intervention, the test cases would not connect to the code delivered by the subject, and the measurement value would not be correct.

Semantic intervention: As the measurement of the FS experiment progressed, we found that the source code needed to be analysed in more detail in order to better understand how the experimental subject solved the experimental task. In this respect, minor changes are not always enough, and

```
public int getfirstThrowValue() {

  return this.firstThrow;
}

public int getSecondThrowValue() {

  return this.secondThrow;
}
```

Fig. 9. Example of source code delivered by an experimental subject for the BSK task.

```
public int getThrow1() {
  return this.firstThrow;
}

public int getThrow2() {
  return this.secondThrow;
}
```

Fig. 10. Example of syntactic intervention on the source code delivered by the subject.

substantial interventions are sometimes required to adapt the code delivered by the subject to the test case class structure and methods. The aim is to get a measurement that evaluates the code implemented by the subject, which does not necessarily adhere to the solution that the researchers had in mind. Developers can come up with surprisingly imaginative and original solutions. It would be unfair to regard such code as invalid just because it does not adopt the expected solution. We refer to these changes as *semantic* intervention because measurers have to understand the code delivered by the subjects, which then has to be carefully connected to the test cases before making the measurement.

In the context of the BSK experimental task, bonus is an extra score added to the bowling game scoreboard. Figure 11 shows the source code delivered by an experimental subject developing BSK. The subject uses two methods to calculate the score: *setBonus(int firstThrow, int secondThrow)* and the *score()* function. In this second option, the bonus is calculated using the bonus variable, which is correct.

However, this is not the only solution for calculating the BSK score. As shown in Figure 12, subjects sometimes develop other types of implementations. In this case, the score is calculated considering the bonus within the frame array instead of the bonus variable as above.

This second option is also valid. In semantic intervention, the measurers identify valid solutions like the above. The activities performed by measurers in order to understand the source code delivered by subjects and make any modifications that are necessary for connection to the test cases are as follows:

—Review the classes of the source code delivered by the subject.
—Review each of the test cases that fail.
—Review the errors with the help of the development environment debugger.

To check the impact of the intervention type, the original code was measured by F. Uyaguari using the three intervention types and by D. Fucci also using the three intervention types.[9] The AH

[9]F. Uyaguari and D. Fucci agree to their identities being published alongside the specific tasks performed in this research. As a result, the measurements made can be said to be reliable, as both are experienced researchers.

```
public void setBonus(int firstThrow, int secondThrow) {
  bonus = new Frame(firstThrow, secondThrow);
}

public int score(){
  int gameScore = 0;
  for(int i = 0; i< frames.size();i++){
    Frame f = frames.get(i);
    int frameScore = f.score();
    if(f.isStrike()){
      if(f.isLastFrame()){
        if (bonus!=null)
        frameScore+=bonus.score();
      }else{
        Frame nf = frames.get(i+1);
        frameScore += nf.score();
        if(nf.isStrike()){
          if(nf.isLastFrame()){
            if (bonus!=null)
            frameScore+=bonus.getFirst();
          }else{
            frameScore+=frames.get(i+2).getFirst();
          }
        }
      }
    }else if(f.isSpare()){
      if(f.isLastFrame()){
        if (bonus != null)
        frameScore+=bonus.getFirst();
      }else{
        Frame nf = frames.get(i+1);
        frameScore+= nf.getFirst();
      }
    }
    gameScore+=frameScore;
  }
  return gameScore;
}
```

Fig. 11. Implementation of the score calculation in the BSK task using a bonus variable.

```
public void setBonus(int firstThrow, int secondThrow) {
  //to be implemented
}

public int score(){
  int result = 0;
  for (Frame f : frames) {
    if (!f.isBonus()) {
      result += f.score();
    }
  }
  return result;
}
```

Fig. 12. Alternative implementation of bonus in the BSK task.

Table 3. Statistical Analysis of the QLTY Response Variable for the Three Intervention Types (Measurer: F. Uyaguari)

	No Intervention	Syntactic	Semantic
(Intercept)	27.57 (6.70)***	63.72 (6.00)***	73.35 (4.75)***
TreatmentTDD-greenfield	49.52 (9.47)***	21.93 (8.48)*	16.15 (6.72)*
R^2	.37	.13	.11
Adj. R^2	.36	.11	.09
Num. obs.	48	48	48

***$p < 0.001$; **$p < 0.01$; *$p < 0.05$.

test case suite was used to make the measurements. These six measurements are new and exclusive to this study, that is, we do not use the measurements reported in Tosun et al. [65].

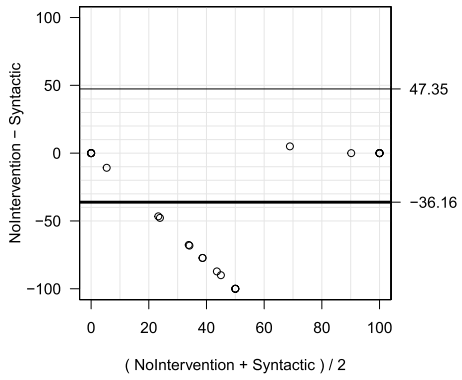
6.2.2 Impact on Statistical Analyses. Table 3 shows the results of the analyses of the *ITL* and *TDD-greenfield* treatments depending on the different intervention types. In this case, the measurements were made by F. Uyaguari. The measurements made by D. Fucci have the same characteristics as the measurements made by F. Uyaguari, as shown in Section 6.2.5 (see Table 4). Albeit less pronounced than for the test cases, the variations in the results of the analyses are still there.

—The effect size differences between *ITL* and *TDD-greenfield* drop as the intervention type becomes less rigorous. There is quite a substantial variation: the effect size of *TDD-greenfield* for the *no intervention* type is 49.52 points, which drops to 16.15 for the *semantic intervention* type.
—The change also affects the statistical significance, that is, the *p*-value increases for the *syntactic* and *semantic intervention* types.

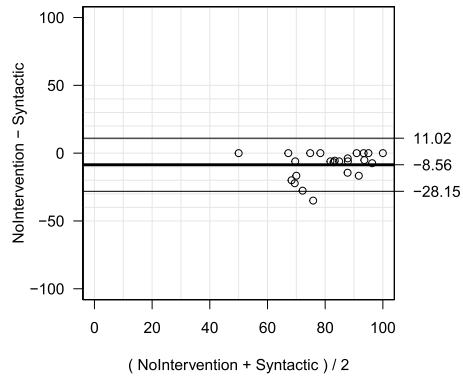
This paper, focusing exclusively on measurement, is not the place to discuss which type of intervention is best. Note, however, that, as Table 3 shows, the experimental results may vary substantially depending on the intervention type used.

6.2.3 Sources of the Differences in the Statistical Analyses. The three possible Bland-Altman plots for the three pairs of intervention types are shown in Figure 13, differentiated, as in Section 6.1, by experimental task (MR or BSK). The differences in the measurements evidenced by the above plots are as follows:

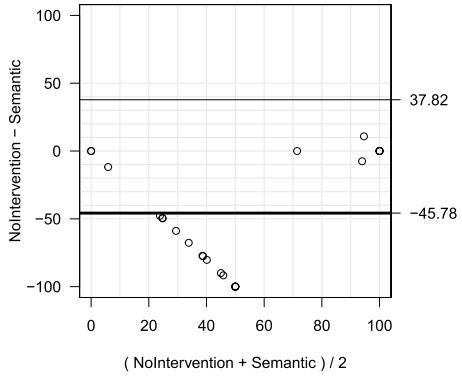
—As before, the biggest differences between the measurements are for the MR task. For BSK, the differences are much smaller.
—The bias or average difference between the measurements made with the *no intervention* type and either of the other *syntactic*/*semantic intervention* types is quite large. Predictably, the difference between *no intervention* and *semantic intervention* is greater than the difference between *no intervention* and *syntactic intervention*, as the differences are proportional to the level of source code manipulation.
—The bias between the measurements after *semantic intervention* and *syntactic intervention* is relatively small for both MR and BSK.
—In any of the three cases, the confidence interval bounds of the difference between the measurements are very wide. This appears to be due to the presence of extreme differences, which, in several cases, can be regarded as outliers and are clearly visible in the Bland-Altman plots shown in Figure 13. MR is more susceptible than BSK to this problem.



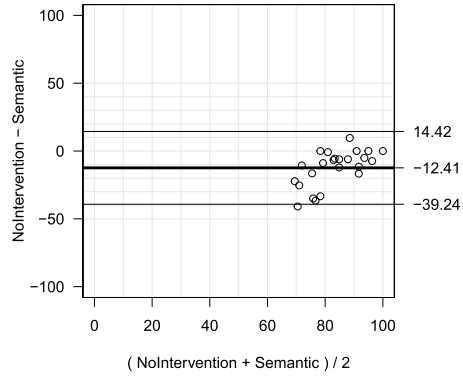
(a) MR



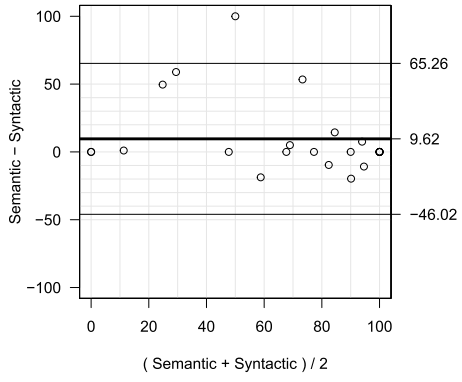
(b) BSK



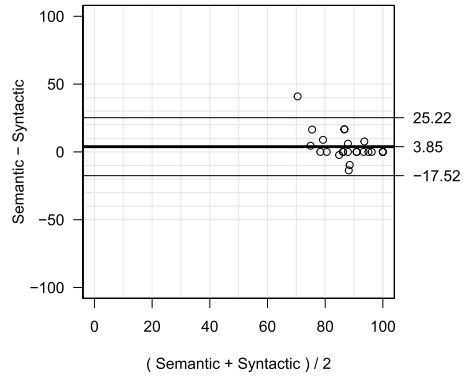
(c) MR



(d) BSK



(e) MR



(f) BSK

Fig. 13. Bland-Altman plots for differences between intervention types (measurer: F. Uyaguari).

Table 4. Analysis of the QLTY Response Variable for the Three Intervention Types (Measurer: D. Fucci)

	No Intervention	Syntactic	Semantic
(Intercept)	25.55 (6.69)***	63.44 (5.59)***	70.66 (4.99)***
TreatmentTDD-greenfield	50.56 (9.46)***	20.53 (7.91)*	14.22 (7.06)*
R^2	.38	.13	.08
Adj. R^2	.37	.11	.06
Num. obs.	48	48	48

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$.

6.2.4 Impact of the Differences on the Statistical Analyses. The differences between the measurements obtained with the different intervention types—no intervention, syntactic intervention and semantic intervention—can be easily mapped to the results of the statistical analyses in Table 3. The reasoning is similar to the arguments reported in Section 6.2.3.

—The average differences between MR and BSK cause the *treatment* to have a smaller effect. For example, comparing no intervention and syntactic intervention, there is a difference of $-36.16 - (-8.56) = -27.6$ points, which corresponds to the difference between the effects 21.93 and 49.52, shown in Table 3.
—The bigger fluctuations in the MR measurements for the *no intervention* and *syntactic intervention* types increase the standard deviation (and could quite possibly reduce the significance of the results).
—However, the p-values would change even if there were no fluctuations in the MR measurement. As the effect sizes drop, the p-values tend to increase, as they depend on the combination of both variance and effect size.

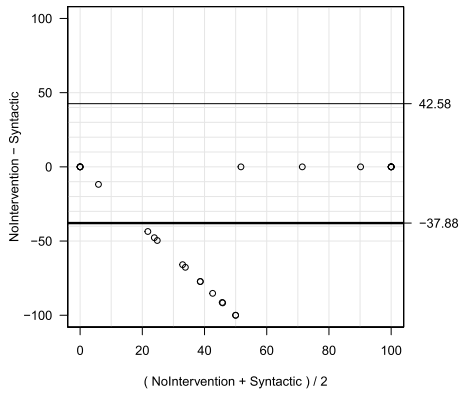
6.2.5 Additional Verification. It might be argued that F. Uyaguari's measurements merely constitute anecdotal evidence. On this ground, D. Fucci, who was the original measurer of the FS experiment, performed the same exercise as F. Uyaguari. Table 4 and Figure 14 replicate the analyses reported for the data obtained by F. Uyaguari shown in Table 3 and Figure 13. It is immediately clear that the results of the statistical analyses and Bland-Altman plots obtained with the measurements made by D. Fucci have the same characteristics as specified above for the measurements made by F. Uyaguari.

6.3 Measurers

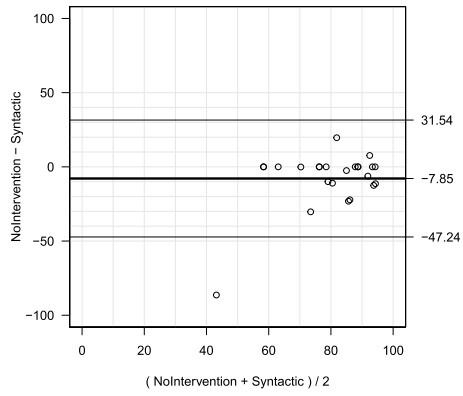
6.3.1 Description of Measurers. The third factor that may influence the measurement process is the measurer. In fact, there are numerous references in the literature to the variability introduced by the measurer [28, 62] to the point that several methods have been developed to calculate the agreement between measurers, published, for example, by Magnusson [41, 42, 49], Farrance [20] and Padoan [51]. In the case of the FS experiment, the measurers were, as already mentioned, F. Uyaguari and D. Fucci.

6.3.2 Impact on Statistical Analyses. The apparent similarity between the measurements by F. Uyaguari and D. Fucci shown in Tables 3 and 4 is confirmed by the statistical analysis of the FS experiment reported in Table 5:

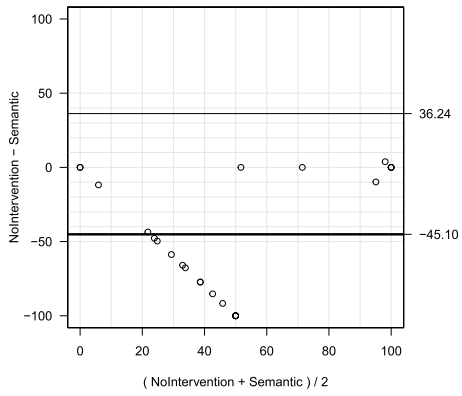
—The *TDD-greenfield* effect is very similar for both measurers. For F. Uyaguari, the effect is 29.20 irrespective of the intervention type applied. For D. Fucci, the effect is 28.44. The difference between the two effects is a negligible 0.72 points.
—The statistical significance is the same in both cases.



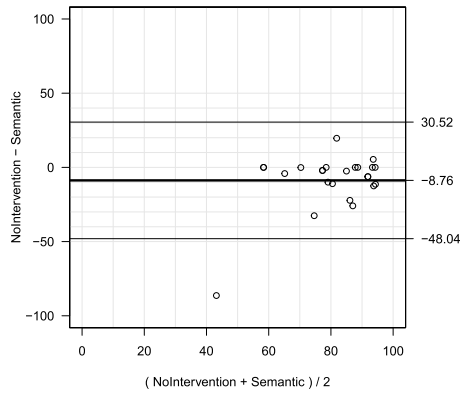
(a) MR



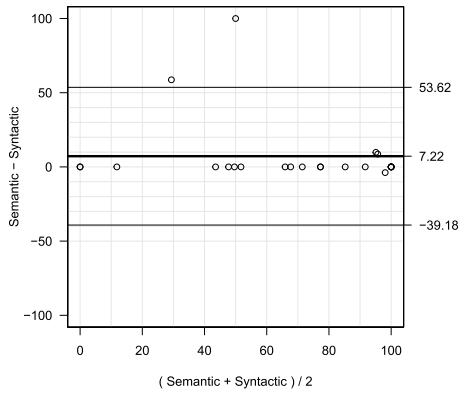
(b) BSK



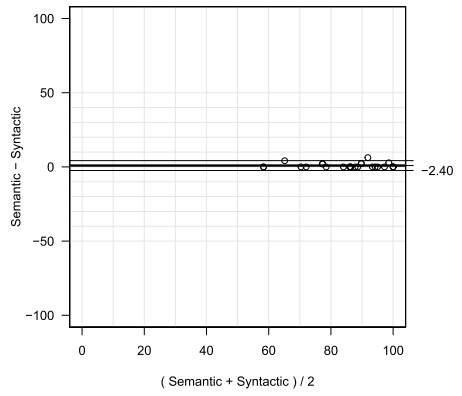
(c) MR



(d) BSK



(e) MR



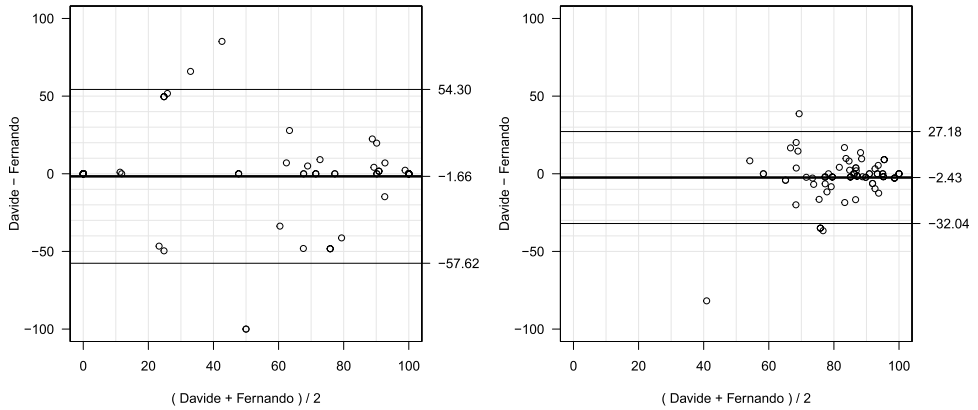
(f) BSK

Fig. 14. Bland-Altman plots for differences between intervention types (measurer: D. Fucci).

Table 5. Analysis of the QLTY Response Variable for the Two Measurers (F. Uyaguari and D. Fucci)

	F. Uyaguari	D. Fucci
(Intercept)	54.88 (3.75)***	53.22 (3.71)***
TreatmentTDD-greenfield	29.20 (5.31)***	28.44 (5.25)***
R^2	.18	.17
Adj. R^2	.17	.17
Num. obs.	144	144

***$p < 0.001$; **$p < 0.01$; *$p < 0.05$.



(a) MR (b) BSK

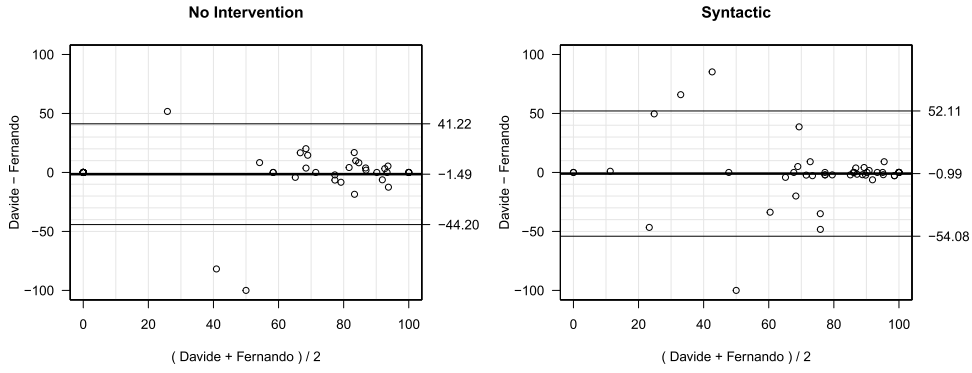
Fig. 15. Bland-Altman plots for differences between measurers.

6.3.3 Source of the Differences. Differences between measurers do not change the value or the sign of the effect, and the statistical significance is unchanged. This suggests that the measurements made on the same code by different measurers are very similar to each other. In the case of TDD experiments that use test cases for quality measurement, the measurer does not appear to be a significant factor of variability. We already mentioned the similarities between the measurements made by F. Uyaguari and D. Fucci in Section 6.3.2. The statistical analyses shown in Tables 3 and 4 are more or less the same, as are the Bland-Altman plots in Figures 13 and 14.

Figure 15 shows a more formal similarity test of the measurements made by the different measurers. Figure 15 groups all the intervention types (we discuss what happens when we divide by intervention type later). The variation bounds are approximately +/− 40 points. This value is not surprising, as the data used in Sections 6.2 and 6.3.3 *ought* to be the same, and the bounds have the same sources (occasional large differences and/or outliers).

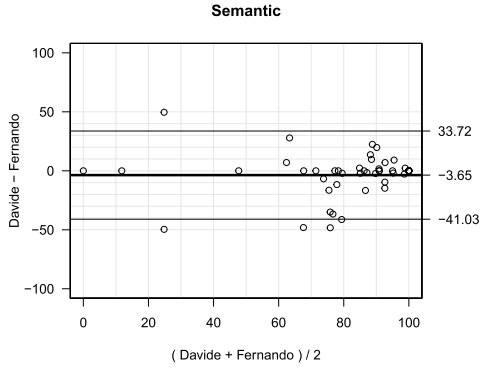
6.3.4 Impact of the Differences on Statistical Analyses. The main result, however, refers to measurer bias. Figure 15 shows that bias is practically non-existent (only −2 points for both measurers), and, additionally, the differences are distributed symmetrically (probably randomly) around the central line. In other words, the measurement carried out by one or other measurer does not influence the results of the TDD experiments that use test cases for measuring quality.

Finally, we should briefly discuss the differences between measurements broken down by intervention type, which are shown in Figure 16. As the differences between MR and BSK are symmetric



(a) No intervention

(b) Syntactic



(c) Semantic

Fig. 16. Bland-Altman plots for differences between measurers, broken down by intervention type.

with respect to the central line, and the bias is very low, we show the Bland-Altman plots for both tasks together. Generally, the Bland-Altman plots in Figures 15 and 16 are very similar. However, there are two aspects that should be highlighted:

—There are differences between measurers when there is *no intervention*. This is noteworthy because a measurer using this intervention type does not manipulate either the subject code or the test cases.
—The less rigorous intervention types do not produce bigger differences than the *no intervention* type. In fact, *semantic intervention* is the intervention type that causes least differences. There is a difference of only +/− 10 points in the bounds of *no intervention* and *syntactic intervention*.

Measurers using *no intervention* measurement do not touch the source code. The differences may be caused by human errors by the measurers when enacting the source code loading procedure, running the test cases and transcribing the measurement results.

7 Systematic Mapping Study

Our aim is to identify the characteristics of the operationalization components (case studies, experimental task, intervention type and measurer) of the TDD experiments that study external

quality and examine how they are reported in order to analyse whether the experimental reports describe the components used to measure the response variable.

7.1 Procedure

The research method used to obtain information about the operationalization of TDD and answer the review research questions described below was an SMS conducted according to Kitchenham and Charters' guidelines [36].

7.1.1 Review Questions. We stated the following review questions:

—Q1. How many TDD experiments study external quality using acceptance tests passed?
—Q2. Is the information with respect to the test cases used in the operationalization of TDD experiments that study external quality available?
—Q3. Is the information with respect to the experimental task used in the operationalization of TDD experiments that study external quality available?
—Q4. Is the information with respect to the code intervention type used in measurement of TDD experiments that study external quality available?
—Q5. Is the information with respect to the measurer of TDD experiments that study external quality available?

The procedure enacted to answer the review questions is described below.

7.1.2 Search Process. For the purpose of answering the review questions, we searched for relevant TDD experiments that study quality published no later than 2021. The search string included the different key words used in the scientific literature. The terms were defined from the following viewpoints:

—Technique: Test Driven Development, TDD, Test-Driven Development, Test First Development
—Importance factor: Quality
—Study types: Experiment

For search string construction, the logical notation was defined as follows: (T1 or T2,..., or Tn) AND (F1 or F2,..., or Fn) AND (E1 or E2,..., or En). The search string was: *("Test Driven Development" OR "TDD" OR "Test-Driven Development" OR "Test First Development") AND "quality" AND "experiment"*.

We used several scientific publisher search engines. The search string was applied to find relevant publications in the online databases used in SE and computer science. The digital databases include IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, most of which were recommended by Brererton et al. [6] for conducting systematic literature reviews in SE.

7.1.3 Study Selection. The study selection was conducted applying a set of inclusion and exclusion criteria. The inclusion criteria are as follows:

—The study must address the TDD development technique,
—The study reports the experimental results,
—The experiment used test cases to measure the external quality response variable.

We excluded studies published in a language other than English. The preliminary database search returned a total of 1922 publications, including conference proceedings papers and journal articles. We then applied the inclusion and exclusion criteria, which output 20 prospective primary studies. Of the 20 experiments, 18 use acceptance test cases passed to calculate external quality. Two experiments were not taken into account on the following grounds: the study by Canfora et al.

[9] measures quality based on the number of implemented test cases, whereas, in the paper by Huang and Holcombe [30], the external client answers a questionnaire to evaluate quality after using the software for a month.

7.1.4 Data Extraction and Analysis Process. From the selected experiments, we extracted the following data:

—Study title
—Year of publication
—Authors
—Study type
—Context (Industry/Academia)
—Control treatment for TDD comparison
—Journal/conference where the study was published
—Experiment result with respect to external quality
—Test cases used
—Author of the test cases
—Test case generation technique
—Name of the experimental task
—Description of the experimental task

Finally, we studied the data extracted to answer the review questions. The results are reported in Section 7.2.

7.2 Results

Figure 17 gives an overview of the identified primary studies. The results are segmented into two separate areas in Figure 17. The first area (left-hand side) basically consists of two XY bubble plots (top and bottom), where the bubbles are placed at the intersections of the following categories: publication type-publication year (top left-hand side) and publication type-information supplied by the experiment (bottom left-hand side). The publication types are conferences and journals. The size of each bubble is determined by the number of primary studies that were classified as members of each category. As the top left-hand side of Figure 17 shows, about two-thirds of the primary studies were published in conferences, which was the preferred medium used by researchers to disseminate their work as of 2012. With respect to the information supplied by the experiment (bottom left-hand side of Figure 17), just under a third of the experiments specify the experimental task and the intervention type. Additionally, none of the 16 TDD experiments that use acceptance test cases passed to measure external quality report the test cases. Although it is possible to figure out which test case generation techniques are used in just under half of the experiments, they are not exactly specified, defined or deducible in the remainder. On the other hand, just over half of the experiments give an account of who made the measurements. The second area (right-hand side) of Figure 17 shows the number of primary studies by year of publication. Looking at this part of Figure 17, half of the experiments on TDD that study external quality were reported in the last 10 years.

7.2.1 TDD Experiments That Study External Quality Using Test Cases. Table 6 shows a list of 18 primary studies. For each of the experiments, it specifies: (i) the primary study identifier (ID); (ii) experimental context (A for academic and I for industrial experiments, respectively), (iii) journal/conference where the primary study was published, (iv) the control treatment for comparison against TDD, (v) the experimental result with respect to quality ((+) the author states that the TDD treatment increases external quality, (−) TDD reduces external quality, (/) there is no

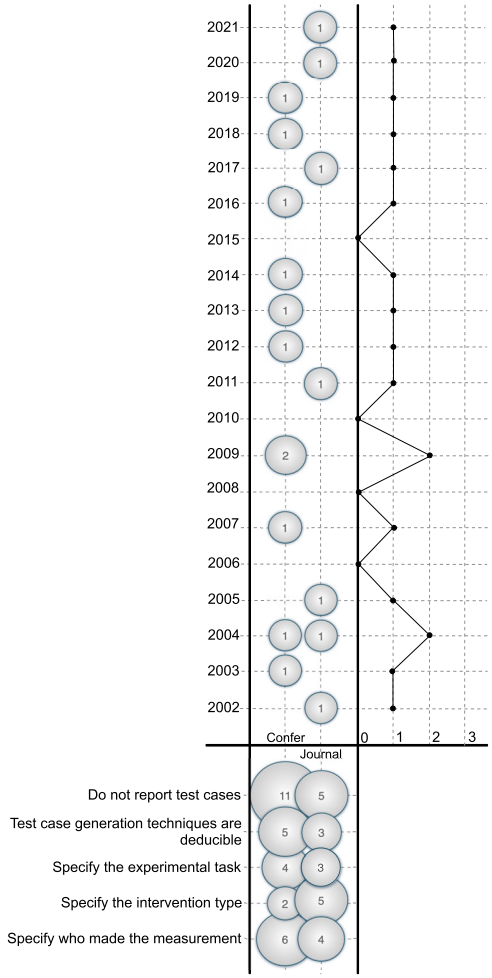


Fig. 17. Mapping with the distribution of the primary studies.

difference between the quality results caused by the treatments), (vi) observations, and (vii) the respective reference.

As observed in Table 6, 11 experiments were run in academia and seven in industry. Of all the experiments, TDD increases external quality in seven (38.8%), decreases external quality in two (11.11%), and there is no difference between TDD and the control treatment in nine (50.0 %). The results are inconclusive and even contradictory.

This study focuses on experiments where quality was measured using acceptance test cases passed. The measurement procedure of these experiments is shown in Figure 4.

7.2.2 Test Cases. Table 7 shows the information extracted from the primary studies on test case generation. For each primary study, it specifies (i) its identifier, (ii) the experimental context, (iii) whether the test cases are reported, (iv) who ran the test cases, (v) the technique used to generate the test cases, and (vi) the respective reference.

As Table 7 shows, test cases are reported in three experiments and not detailed in 15, that is, 16.66% and 83.33%, respectively. The authors fail to specify who generated the test cases in seven

Table 6. TDD Experiments that Study External Quality

ID	Industry/ Academia	Journal/Conference	Control	Experiment results according to author	Observations	Reference
[PS11]	A	EASE 2012	TL	(+) Quality is better with TF than TL.	It is not statistically significant.	[11]
[PS19]	A	SIGSE 2009	TL	(/) There is no significant difference between subjects applying TL vs. TDD.		[15]
[PS25]	A	IEEE Transactions on Software Engineering	ITL	(/) There is no significant difference between ITL & TDD. ITL: 85% quality, TDD: 83% quality, p-value: 0.25		[19]
[PS30]	A	ESEM 2013	ITL	(/) Small effect size: 0.11, p-value: 0.53	The null hypothesis is not rejected.	[23]
[PS108]	A	ESEM 2016	ITL	(/) Very small difference between TDD & ITL wrt quality	The difference is not statistically significant.	[22]
[PS37]	I	Information and Software Technology	Waterfall	(+) TDD increases quality by 18% over waterfall.		[26]
[PS38]	I	METRICS 2004	TL	(+) TDD increases quality wrt TL.		[27]
[PS40]	A	ESEM 2007	Waterfall	(/) Quality increased due to testing effort not to development technique.		[29]
[PS78]	A	IEE Proceedings - Software	Waterfall	(/) TDD does not increase program reliability.	Reliability is measured with acceptance cases passed.	[44]
[PS80]	I	EASE 2014, ACM	ITL	(/) Similar number of acceptance tests passed.	The difference is not statistically significant.	[46]
[PS82]	A	Eurocon 2003	ITL	(−) Small improvements for ITL. TDD: mean value of 92.6/120 acceptance tests passed; ITL: 95.1/120	There is no statistically significant difference.	[53]
[PS83]	A	Information and Software Technology	Micro ITL	(+) The TDD effect is small and positive wrt the percentage of acceptance test cases.	There is no statistically significant difference.	[52]
[PS98]	A	ITNG	ITL	(+) ITL has 39% more defects than TDD (number of recorded defects).		[67]
[PS115]	I	ICCIKE 2019	BDD	(+) TDD increases external quality.		[18]
[PS116]	A	IEEE Transactions on Software Engineering	TL	(+) TDD has an edge over TLD in terms of code quality (1.8 fewer bugs).	It is statistically significant.	[54]
[PS117]	I	APSEC 2018	YW & ITL	(/) TDD slightly outperforms YW & ITL.	It is not statistically significant.	[56]
[PS118]	I	Empirical Software Engineering	ITL	(/) There is no statistically significant difference between the quality of the work performed by subjects for both treatments.		[65]
[PS119]	I	IEEE Transactions on Software Engineering	ITL	(−) The subjects produce better quality code with ITL than with TDD where the subtasks are divided into user stories, coded and tested.		[66]

TF = Test First; TL = Test Last; ITL = Iterative Test Last, BDD = Behaviour-driven development; YW = Your way.

experiments (38.88%). No information about the technique used to generate the test cases is supplied in 10 experiments (55.55%). None of the 18 experiments explicitly report the name of the technique used to generate the test cases that are used as an instrument for measuring quality.

Based on the analysis of the primary studies, we find that there are differences between the case studies and, generally, between the instruments that the researchers use in the experiments to measure the same constructs. The percentage differences appear to be small, but, in actual fact, we deduced most of the information shown in Table 7. The authors provide little information on the characteristics of the test cases used in the measurements. Additionally, they offer little or no information on how the test cases were generated, where they were obtained or who created them. None of the analysed experiments explicitly explain whether they use a specific test case generation technique. Eight of the analysed experiments (44.44%) use AH, which is essentially tantamount to the researcher's best judgement, instead of a formal test case generation technique.

7.2.3 Experimental Task. The experimental task is another focus of this study. In the 18 experiments that study quality, we identified the name of the task used and determined whether or not the author reports the task specification.

As Figure 4 shows, the experimental task is another important operationalization component in TDD. Table 8 reports the characteristics of the experimental task used in each of the 18 primary studies. For each of these experiments, it specifies (i) their identifier (ID), (ii) the experimental context, (iii) whether or not the BSK experimental task is used, (iv) the name of the experimental

Table 7. Test Cases in TDD Experiments

ID	Industry/ Academia	Are the test cases reported?	Who generated the test cases?	Test case generation technique used	References
[PS11]	A	NO	We deduced that it was the subjects.	We deduced that technique is AH.	[11]
[PS19]	A	NO	The instructor created the test case suite.	We deduced that technique is AH.	[15]
[PS25]	A	NO	Black box acceptance tests were written by the researcher	Unspecified.	[19]
[PS30]	A	NO	They were created by the researcher.	We deduced that technique is AH.	[23]
[PS108]	A	NO	We deduced that they were created by the researcher.	We deduced that technique is AH.	[22]
[PS37]	I	YES, in the thesis [25] but NOT in the primary study	They were created by the researcher.	We deduced that technique is AH.	[26]
[PS38]	I	NO	There are developer test cases created by the developer and client test cases created by the researcher.	Unspecified.	[27]
[PS40]	A	NO	They were created by the researcher.	Unspecified.	[29]
[PS78]	A	NO	Unspecified	We deduced that technique is AH.	[44]
[PS80]	I	NO	Unspecified	Unspecified.	[46]
[PS82]	A	NO	Unspecified	We deduced that technique is AH.	[53]
[PS83]	A	NO	They were created by the researcher.	We deduced that technique is AH.	[52]
[PS98]	A	NO	Unspecified	Unspecified.	[67]
[PS115]	I	NO	Unspecified	Unspecified.	[18]
[PS116]	A	YES, in the replication package	We deduced that they were created by the authors.	Unspecified.	[54]
[PS117]	I	NO	Unspecified	Unspecified.	[56]
[PS118]	I	NO	Unspecified	Unspecified.	[65]
[PS119]	I	YES	They were created by the researchers.	Unspecified.	[66]

task, (v) whether or not it reports the experimental task specification, and (vi) the respective reference.

According to Table 8, three experiments (16.66%) fail to specify the name of the experimental task, and 11 experiments (61.11%) do not report the experimental task specification. Bowling Scorekeeper (BSK)[10] was the experimental task used in 10 experiments (55.55%). Five experiments used a task other than BSK (27.77%). Only three primary studies (16.66%) explicitly report the experimental task specification used in the experiment and three primary studies (16.66%) reference other documents that are available in a repository containing the specification.

7.2.4 Intervention Type. When the measurer applies the test cases to obtain the value of the response variable, the source code of the experimental task developed by the subjects must be connected to the test cases. Source code intervention is often necessary, which is carried out at the measurer's discretion. We gathered information on the type of intervention carried out by the measurer on the source code during measurement in the 18 experiments that study quality.

[10]Bowling Scorekeeper by C. Martin [43].

Table 8. Experimental Task in the TDD Experiments That Study External Quality

ID	Industry/ Academia	Does it use BSK?	Name of experimental task	Does it report the specification of the experimental task?	References
[PS11]	A	YES	BSK	NO	[11]
[PS19]	A	Unspecified	It mentions, but does not specify, 7 projects.	NO	[15]
[PS25]	A	YES	BSK	NO	[19]
[PS30]	A	YES	BSK	NO	[23]
[PS108]	A	YES	BSK	NO	[22]
[PS37]	I	YES	Bowling Game (similar to BSK)	YES, in the thesis	[26]
[PS38]	I	NO	Program A – Registering a new project Program B – Recording time against a project	YES	[27]
[PS40]	A	NO	Student registration system / Simple ATM system	YES	[29]
[PS78]	A	NO	GraphBase, related to graphs	NO	[44]
[PS80]	I	YES	BSK	YES, on the experiment web page	[46]
[PS82]	A	Unspecified	Unspecified	NO	[53]
[PS83]	A	NO	Distributed database server with built-in data replication mechanism, implementation of a chat server	NO	[52]
[PS98]	A	Unspecified	Unspecified	NO	[67]
[PS115]	I	YES	String calculator, BSK	NO	[18]
[PS116]	A	NO	1) Propose a data structure to hold information about daily gas flow for every hour in local time. 2) Assuming all the required data for the structure designed in Task 1 is stored in a text file, propose a method of verifying its correctness. The participants assume the availability of a method GetNumberOfHoursInDay, which returns an integer and accepts a single DateTime parameter.	Partially	[54]
[PS117]	I	YES	BSK, MR, Spreadsheet	YES, in a link	[56]
[PS118]	I	YES	MR API, BSK, MusicPhone	NO, outlines tasks	[65]
[PS119]	I	YES	MR API, BSK	YES	[66]

The last but one column in Table 9 shows the type of source code intervention in each of the 18 primary studies. Code intervention was not reported in 11 experiments (61.11%), there was no source code intervention in two experiments (11.11%), the subjects in three experiments (16.66%) modified the source code after running the test cases in order to correct errors, and the researcher of one experiment (5.55%) stated that changes were made so that the code executed successfully.

7.2.5 Measurers. The measurer applies the test cases to obtain the value of the response variable. Table 9 specifies for each primary study: (i) its identifier (ID), (ii) the experimental context, (iii) the measurer, (iv) data on the source code intervention when making the measurements, and (v) the

Table 9. Measurer and Type of Code Intervention in TDD Experiments That Study External Quality

ID	Industry/ Academia	Who made the measurements?	Source code intervention during measurement	References
[PS11]	A	Unspecified	Unspecified	[11]
[PS19]	A	Researchers	Unspecified	[15]
[PS25]	A	Unspecified	Test cases were run automatically at the end of the experiment. We deduced that there was no source code intervention.	[19]
[PS30]	A	Unspecified	Unspecified	[23]
[PS108]	A	Researcher (FU)	Unspecified	[22]
[PS37]	I	Unspecified	Unspecified	[26]
[PS38]	I	Subjects	The test cases are run automatically. We deduced that there was no source code intervention.	[27]
[PS40]	A	Subjects	The test cases were run, and the errors were observed and corrected.	[29]
[PS78]	A	Subjects	The subjects corrected the defects.	[44]
[PS80]	I	Researchers	Unspecified	[46]
[PS82]	A	Unspecified	Unspecified	[53]
[PS83]	A	Subjects	The subjects modified the code until the test case pass rate was 100%.	[52]
[PS98]	A	Committee composed of representatives of subject groups	Unspecified	[67]
[PS115]	I	Unspecified	Unspecified	[18]
[PS116]	A	Measured automatically using web services at the end of each iteration	Unspecified	[54]
[PS117]	I	Unspecified	Unspecified	[56]
[PS118]	I	Unspecified	The measurement description does not state that there is code intervention.	[65]
[PS119]	I	Researcher	Changes were made so that the code ran successfully without interfering with the code delivered by the subject.	[66]

respective reference. According to Table 9, we find that eight experiments (44.44%) do not specify who does the measuring, four experiments (22.22%) identify the researchers as the measurers, the subjects did the measuring in four experiments (22.22%), a committee was formed to measure quality in one experiment (5.55%), and a weekly measurement was made automatically at the end of each iteration in one experiment (5.55%).

7.3 Review Conclusion

With respect to the analysed measurement process components, such as test cases used to measure external quality, experimental task, intervention type and measurers, the researchers do not report or only partially report the characteristics of these components. We find that the characteristics of the instruments—experimental task, development environment, test case generation, test case generation technique, measurers, code interventions—used by researchers in each experiment may

well differ. Although there are SE experiment reporting guidelines, researchers do not include all the information in experiment reports.

8 Validity Threats

This section discusses the validity threats with respect first to the quantitative research and then to the qualitative study. With regard to the quantitative research, we take into account the following. For the purpose of assuring *conclusion validity*, we took the following three precautions. First, we used several methods of comparison, namely the concepts established in ISO 5725 [31] and Bland-Altman plots [5]. Second, the conclusions are based on a total of 288 measurements made for this study. Third, data capture and analysis were largely automated to prevent problems of transcription and manipulation. R and Sweave were used to make and integrate all the calculations.

With regard to *internal validity*, three measures were taken to ensure that the conclusions capture the effect of the independent variable on the dependent variable. First, the measurements were taken independently, based on the original source code created by the experimental subjects, that is, code was not reused to rule out undesired dependencies across measurements. Second, the measurers (i.e., F. Uyaguari and D. Fucci) acted independently without communicating with each other. Finally, the data used in this study are not the first measurements made by the measurers F. Uyaguari and D. Fucci. These measurers had already made measurements as part of other experiments.

To assure that the resulting conclusions are linked to the independent and dependent variable operationalizations and not to underlying constructs, we took two precautions with respect to *construct validity*. First, in this study, we studied differences between the measurements of the quality response variable using test cases only. Second, for each measurement method component, that is, the independent variables, we study three different variants: (1) two types of test case generation strategies; (2) two different measurers; (3) three intervention types.

To ensure that the research results can be generalized to other populations, situations or moments of time, we took three precautions with respect to *external validity*. First, to increase the possibility of generalizing results, we measured experiments carried out by professional subjects that are representative of industry (FS). Second, we used representative test case suites created by different people. The first AH test suite was used to measure experiments in industry within the ESEIL project [35]. The second suite was generated as part of a master thesis using the EP technique [59]. Third, the measurements were made by representative measurers: two PhD students (at measurement time) with experience in the software area.

With respect to the qualitative part, note that this study focused on TDD. The research method used was a SMS. In order to increase the study validity, we identified relevant experiments on TDD within a range of databases like IEEE Xplore, ACM Digital Library, SpringerLink and ScienceDirect used to answer the research questions.

In order to carry out a good quality SMS, the research was conducted by a team of researchers. The search string, primary study selection and data extraction was carried out by the researcher F. Uyaguari and reviewed by another three of the paper authors to ensure that the procedure complied with Kitchenham and Charters' guidelines. The results were synthesized by all four researchers through negotiation.

Finally, there are several frameworks for evaluating the quality of systematic reviews. We used the evaluation criteria proposed by Thompson et al. [63] because they are domain independent and easy to use. Table 10 reports the results of the evaluation of the study, which concludes that the literature review results are reliable.

Table 10. Evaluation Criteria Used for the Systematic Review

Evaluation criteria	Evaluation of review questions
Was the bibliographic search exhaustive?	Yes, although Scopus and WoS were missing, the review includes the four major publishers of SE articles (IEEE, ACM, SpringerLink and Elsevier).
Were the criteria used to select articles for inclusion appropriate?	Yes, we used appropriate criteria for searching for articles that report the results of experiments on TDD that study external quality using acceptance tests passed.
Were the studies included valid enough to answer the stated research questions?	Yes, only controlled experiments were selected.
Were the results similar from one study to another?	Yes, they are articles that are reported in sufficient detail and published in journals/conferences where supplementary information is included in web appendices.

9 Discussion

9.1 Main Results of the Quantitative Analysis

The results reported in Section 6 highlighted that the measurement process components (test suites, intervention types and measurers) have a powerful influence on the results of both measurements and statistical analyses.

Firstly, let us discuss which measurement method components may threaten the validity of the statistical analyses of the experiments, which was one of the primary aims of this study. With regard to TDD, which is the area explored by the experiments used in the analyses, our observations suggest that: (1) the test suites have a major impact on both the measurements and the statistical analyses of the experiments, (2) the intervention type has an impact on the measurements, and a more limited impact on the statistical analyses of the experiments and (3) the measurers have an impact on the measurements but do not influence the statistical analyses of the experiments.

The more positive measurements yielded by AH are the source of the differences between the measurements made using different test cases. They have an impact on the influence of the treatment (TDD) to the point that they change the sign of the effect, confirming the results obtained by Dieste et al. [17].

With respect to intervention type, the impact on the statistical results is reflected by the fact that, as source code intervention increases, the treatment effect decreases with respect to both the response variable value and statistical significance. The following example illustrates the above statement. We know that syntactic intervention is a more rigorous intervention type than semantic intervention, which means that the QLTY values are lower in the first than in the second case. This is illustrated to perfection in Figures 13 and 14, which compare the two intervention types. This variation modifies the effect on the statistical analysis because all the codes, irrespective of whether they were developed with ITLD or TDD or simply MR or BSK, are equally affected by low QLTY values. While, unlike the test cases, the sign of the effect is unchanged, it is important to bear this issue in mind to standardize the measurement procedure that is being applied to operationalize the experiments.

The conclusions of the experiment are unchanged irrespective of whether the results of an experiment are measured by one or other measurer or the code delivered by the subjects is unmanipulated or manipulated differently for the purpose of connection to the measurement test suite.

Caution should be exercised when generalizing. However, there are some recommendations that are probably generally applicable:

—Measurement using test suites is largely automated. It is true that the measurers have to connect the code developed by subjects to the test suites using a series of predefined intervention types. Once this step is completed, however, measurement is made automatically. This process is largely equivalent to measurement in chemistry: the sample is prepared for measurement, and special measurement apparatus is immersed in the solution. The intervention types (equivalent to sample preparation in chemistry) are somewhat but not overly subjective. Therefore, we recommend that, irrespective of the particular branch of science, the measurement method must, whenever possible, be automated.

Measurement is automated in TDD experiments by using test suites. However, as reported in Section 6, test suites are the most troublesome measurement method component because they behave differently with respect to AH and EP. The question then is, can these behavioural differences be avoided? In principle, we believe that they can be avoided in two different ways. First, performing a pilot experiment, which is capable of testing measurement reproducibility [17] and, if necessary, adjusting test suites. Second, running a reproducibility analysis using reference code. This code could be composed of variants of the MR and BSK tasks that implement some, but not all, user stories. The second option appears to be more preferable on three grounds: (1) it requires less effort; (2) it assures more control over measurements; (3) the comparison of two measurements is unnecessary, as a reference value (QLTY associated with the reference code) is available. In other words, the measurement instrument (test suites) should not be generated based exclusively on the fundaments of theory but must be empirically validated using reproducibility analysis [17].

—One way of improving measurement uniformity is, of course, to take the subjectivity out of the application of the intervention types. As discussed in Section 6 above, the no intervention type is dysfunctional and, as such, is not an option. We do not know whether this applies in other areas, but it is highly likely. One credible simile are student examinations. For example, have you never hastily marked a student's exercise 0 and then, on closer reading, found it to be satisfactory or even brilliant? Alternatively, have you never rejected an article during a peer review process, which was finally accepted? A cursory examination does not appear to be the best possible evaluation strategy.

In the field of TDD, one possible way of eliminating measurer bias would be to force subjects to respect the experimental task APIs. Intervention is required primarily due to API modifications. However, it is questionable whether you can oblige a subject to respect an API with which they disagree. It would perhaps be preferable to limit measurer access to the code delivered by the experimental subjects, giving them access to the API alone. This could reduce possible decision-making bias.

Generally, we recommend the use of blinding. The measurers should not have any preference with respect to the code delivered by the subjects (or any other product resulting from the experimental sessions), the applied treatment or, even the measurement results. This would minimize measurement subjectivity and bias.

—It was determined in Section 6 that statistical analysis was not affected when the measurement method components behave consistently with respect to all the factor levels. One much simpler way of testing whether the statistical analysis was affected would be to run the analyses using different measurements and compare the results. However, we do not advise the experimenters to make post hoc decisions. Any decision should precede the statistical analysis of the results (except, of course, the choice of the method (AIC, BIC, etc.) to study model fit).

Different test case suite and measurer behaviour with respect to the MR and BSK tasks is a source of problems with measurements and measurement analysis. The use of two tasks is only necessary in within-subjects experiments, where the subjects apply more than one treatment to different tasks. In the case of between-subjects experiments, where there is only one experimental task, behaviour is bound to be the same, and test suites and intervention type have no harmful effects.

9.2 Main Results of the Qualitative Study

As observed in the scientific literature, TDD experiments that study quality repeatedly yield different, and sometimes even contradictory, results (for example, George and Williams [26] claim that TDD favours an 18% quality increase with respect to the waterfall method, whereas Huang and Holcombe [30] claim that there is a very small difference, which is not statistically significant). In some secondary studies [4, 10, 55], the researchers look for explanations in the experimental context (industry or academia) and study rigour [45]. However, these studies fail to take into account other factors, such as components relative to the instrumentation, which could be having an effect on the experimental results.

Researchers only partially report the instrumentation of both the test cases used to measure the external quality and the experimental task. We found that different researchers may very well use instruments—experimental task, test cases used for measurement, test case generation technique, etc.—with different characteristics in their experiments. Even though they have access to guidelines and recommendations for reporting experiments in SE, the authors do not include all the information in the experiment reports. The authors tend to partially report instrumentation-related issues in TDD experiments.

If the researchers do not report the information on the experiments in full, it is impossible to evaluate the external validity of the results or deduce what influence this may have on the experimental results. Different instrument characteristics will probably lead to different experimental results, which are being overlooked by researchers. Access to more detailed information about the test cases and experimental task, code intervention type and other components would be useful for studying the influence of instrumentation on the experimental results.

Additionally, the failure to record these measurement process components is an obstacle to replication, and replication underpins the formation of families of experiments to obtain more reliable results. If we had information on measurement process operationalization, it would help effective experiment replication, assuring that the replication results were not influenced by the factors observed in this study.

10 Conclusions

During the experimental process, the operationalization of the response variables of the experiments is highly dependent on the researcher. Researchers operationalize each in their own way, and there are no protocols or recommendations about how to carry out the operationalization process. We addressed this problem by conducting a mixed methods study that has highlighted several complexities related to measurement.

From the operationalization process viewpoint, the FS experiment is no different from other TDD experiments available in the literature. As there was no experiment measurement process, the first step of our research was to define a protocol based on the procedures used in other experiments run. This measurement procedure was framed within SE measurement theory [21].

We identified a set of relevant measurement method components, such as test suites, development environment, measurers, source code intervention type, metric, test framework, code compilers, etc. We selected and studied the components that we believed to have most influence on

the measurement results: test suites [17], measurers and source code intervention based on the experience acquired during the execution of the ESEIL experiment [35].

The biggest problem that we detected concerns the test suites. In principle, we thought that the source of the problem was AH test case creation, which a more formal, EP-based test generation method would solve. However, we were wrong. On the one hand, no test suite creation method is intrinsically bad; the question is whether the test suite matches the expectations of the researchers and experimental subjects. If the researchers and experimental subjects do not expect code exceptions to be explicitly controlled, an EP-based test suite will return much poorer results than an AH test suite generated overlooking the exceptions. On the other hand, test suites tend not to respect a basic metric property known as *representation condition* [21]: codes with more functionalities should return proportionally higher QLTY values. This is not generally the case with an EP-based test suite.

We were surprised to find that the measurers had little impact and intervention types, a moderate influence on statistical analyses. In truth, we expected the exact opposite. This takes the pressure off the measurers, as it does not appear to be necessary to establish specific controls to assure the validity of the data gathered by different measurers who somehow manipulate the code delivered by the experimental subjects.

After reviewing the related scientific literature, we found that most experimental papers do not offer detailed information on how the response variables were operationalized. One especially important aspect is that is missing from most experimental reports is the measurement process. For example, TDD experiments do not indicate how the measurement test suites were generated. Likewise, these test suites are not usually available for independent verification. Predictably then, other details present in many TDD (and generally SE) experiments, such as intervention type and measurer, are, in many cases, not even mentioned. All this means that there are important factors that are not being taken into account and may be having an influence on the experimental results.

Our findings suggest that more attention needs to be paid to the response variable operationalization process. It is not possible to give instructions on how to define the response variables, as they are highly domain dependent. However, once the response variable has been defined, two precautions should be taken before it is used. The first is to run trials, studying measurement repeatability and reproducibility ([17] provides an adequate introduction for SE researchers in this regard). Such trials can also be useful for developing the measurement protocol and anticipating possible scenarios (like the intervention type). The second is to check that the response variable definition has the properties of a measurement in the mathematical sense of the term (in this respect, see [21, Chapter 2]). In particular, the metric should satisfy the *representation condition*, which is by no means straightforward. At the very least, it is necessary to measure different objects (in this case, code implementing the experimental tasks) and check that the objects that have a greater "quantity" of the property of interest (in this case, code functionality) return higher metric values (in this case, greater external quality values). A substantial number of objects may be required to reliably run this analysis.

Measurement has been found to be a much less critical aspect than response variable definition. However, this may be true only under certain circumstances, such as measurement with use cases, whereas, in others, e.g., a measurement made by a judge using a checklist, measurement could be trickier. In any case, we believe that two important actions should be taken. First, measurement should, as far as possible, be automated. This has many advantages: greater measurement repeatability and reproducibility, less bias, and faster output of raw data. Second, if measurement cannot be fully automated, it may be worthwhile applying blinding to prevent measurer bias. For example, researchers recall having invested more effort in testing the code of people that they considered to

be capable programmers than other participants, which could have boosted their quality levels. Blinding assures that the results of the measurement are not affected by such biases.

Finally, researchers must take special care with measurement protocol reporting. This protocol should be accompanied by the measurement instruments, which are necessary for experimental reanalysis and replication. We venture to suggest that both the measurement instruments and the measurement environment should be made public. In the case of the FS experiment, for example, the measurement environment would be the projects developed by the experimental subjects, plus the measurement test cases adequately connected with the code by means of some sort of intervention. Nowadays, we have sophisticated tools such as version control systems that can be used to run `diffs` to check the changes made by measurers.

This research leaves a lot of open questions, which we were not able to address on the grounds of time and resources. We list below just four important future lines, some of which we hope to take up in the near future. First, the recommendations that we have provided were inferred from the results reported here. However, as this research clearly shows, inferences are fallible. To increase the robustness of the findings of this research, this study needs to be repeated on other experiments using different quality response variables. Second, it remains to determine how an experimenter can decide whether a measurement instrument (for example, AH test suite or EP test suite) is reliable. The steps of the measurement instrument reproducibility analysis should also be protocolized. Third, we recommended blinding to reduce measurer bias when applying different intervention types. However, another path for improving the validity of the measurements would be to use different measurers, intervention types and even test suites and average the results using either simple or weighted means. Finally, we plan to analyse other experimental instruments, such as development environment, measurer learning and measurement protocols, which are not mentioned in this study and could also influence the results.

References

[1] Alain Abran, Asma Sellami, and Witold Suryn. 2003. Metrology, measurement and metrics in software engineering. In *Proceedings of the 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No. 03EX717)*. IEEE, 2–11. DOI: https://doi.org/10.1109/METRIC.2003.1232451

[2] Kent Beck. 2003. *Test-Driven Development: By Example*. Addison-Wesley Professional.

[3] Thirumalesh Bhat and Nachiappan Nagappan. 2006. Evaluating the efficacy of test-driven development: Industrial case studies. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06)*. ACM, New York, NY, 356–363. DOI: https://doi.org/10.1145/1159733.1159787

[4] Wilson Bissi, Adolfo G Serra, and Maria C. Figueiredo. 2016. The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology* 74 (2016), 45–54. DOI: https://doi.org/10.1016/j.infsof.2016.02.004

[5] J. Martin Bland and Douglas G. Altman. 1986. Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet* 327, 8476 (1986), 307–310. DOI: https://doi.org/10.1016/S0140-6736(86)90837-8

[6] Pearl Brereton, Barbara A Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* 80, 4 (2007), 571–583. DOI: https://doi.org/10.1016/j.jss.2006.07.009

[7] SIGIST British Computer Society. 2001. *Standard for Software Component Testing*. Technical Report Working Draft 3.4. British Computer Society (BCS) Specialist Interest Group in Software Testing (SIGIST). Retrieved from https://testingfreak.files.wordpress.com/2008/12/component-testing.pdf

[8] Mario Bunge. 1999. *Buscar la filosofía en las ciencias sociales*. Siglo XXI, Buenos Aires, Argentina.

[9] Gerardo Canfora, Aniello Cimitile, Felix Garcia, Mario Piattini, and Corrado Aaron Visaggio. 2006. Evaluating advantages of test driven development: A controlled experiment with professionals. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06)*. ACM, New York, NY, 364–371. DOI: https://doi.org/10.1145/1159733.1159788

[10] Adnan Causevic, Daniel Sundmark, and Sasikumar Punnekkat. 2011. Factors limiting industrial adoption of test driven development: A systematic review. In *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation*. IEEE, 337–346. DOI: https://doi.org/10.1109/ICST.2011.19

[11] Adnan Causevic, Daniel Sundmark, and Sasikumar Punnekkat. 2012. Test case quality in test driven development: A study design and a pilot experiment. In *Proceedings of the 16th International Conference on Evaluation Assessment in Software Engineering (EASE)*. IET, 223–227. DOI: https://doi.org/10.1049/ic.2012.0029
[12] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, New York.
[13] John W. Creswell and David Creswell. 2018. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (5th ed.). SAGE Publications, London, UK.
[14] Bill Curtis. 1980. Measurement and experimentation in software engineering. *Proceedings of the IEEE* 68, 9 (1980), 1144–1157. DOI: https://doi.org/10.1109/PROC.1980.11813
[15] Chetan Desai, David S. Janzen, and John Clements. 2009. Implications of integrating test-driven development into CS1/CS2 curricula. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. ACM, New York, NY, 148–152. DOI: https://doi.org/10.1145/1508865.1508921
[16] Oscar Dieste, Alejandrina M Aranda, Fernando Uyaguari, Burak Turhan, Ayse Tosun, Davide Fucci, Markku Oivo, and Natalia Juristo. 2017. Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study. *Empirical Software Engineering* 22, 5 (2017), 2457–2542. DOI: https://doi.org/10.1145/3202710.3203163
[17] Oscar Dieste, Fernando Uyaguari, and Natalia Juristo. 2021. Test cases as a measurement instrument in experimentation. arXiv:2111.05287. Retrieved from https://arxiv.org/abs/2111.05287
[18] Avishek Sharma Dookhun and Leckraj Nagowah. 2019. Assessing the effectiveness of test-driven development and behavior-driven development in an industry setting. In *Proceedings of the International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. IEEE, 365–370. DOI: https://doi.org/10.1109/ICCIKE47802.2019.9004328
[19] Hakan Erdogmus, Maurizio Morisio, and Marco Torchiano. 2005. On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering* 31, 3 (2005), 226–237. DOI: https://doi.org/10.1109/TSE.2005.37
[20] Ian Farrance, Tony Badrick, and Robert Frenkel. 2018. Uncertainty in measurement and total error: different roads to the same quality destination? *Clinical Chemistry and Laboratory Medicine* 56, 12 (2018), 2010–2014. DOI: https://doi.org/10.1515/cclm-2018-0421
[21] Norman Fenton and James Bieman. 2014. *Software Metrics: A Rigorous and Practical Approach*. CRC Press, Boca Raton, FL.
[22] Davide Fucci, Giuseppe Scanniello, Simone Romano, Martin Shepperd, Boyce Sigweni, Fernando Uyaguari, Burak Turhan, Natalia Juristo, and Markku Oivo. 2016. An external replication on the effects of test-driven development using a multi-site blind analysis approach. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*. ACM, New York, NY, Article 3, 10 pages. DOI: https://doi.org/10.1145/2961111.2962592
[23] Davide Fucci and Burak Turhan. 2013. A replicated experiment on the effectiveness of test-first development. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 103–112. DOI: https://doi.org/10.1109/ESEM.2013.15
[24] Davide Fucci, Burak Turhan, and Markku Oivo. 2014. Impact of process conformance on the effects of test-driven development. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. ACM, New York, NY, Article 10, 10 pages. DOI: https://doi.org/10.1145/2652524.2652526
[25] Body George. 2002. *Analysis and Quantification of Test-Driven Development Approach*. Master's Thesis. North Carolina State University, North Carolina State University. Retrieved from http://www.lib.ncsu.edu/resolver/1840.16/2431
[26] Boby George and Laurie Williams. 2004. A structured experiment of test-driven development. *Information and Software Technology* 46, 5 (2004), 337–342. DOI: https://doi.org/10.1016/j.infsof.2003.09.011
[27] Adam Geras, Michael R. Smith, and James Miller. 2004. A prototype empirical evaluation of test driven development. In *Proceedings of the 10th International Symposium on Software Metrics*. IEEE, 405–416. DOI: https://doi.org/10.1109/METRIC.2004.1357925
[28] Peter Gibbons, Charlotte Dumper, and Cameron Gosling. 2002. Inter-examiner and intra-examiner agreement for assessing simulated leg length inequality using palpation and observation during a standing assessment. *Journal of Osteopathic Medicine* 5, 2 (2002), 53–58. DOI: https://doi.org/10.1016/S1443-8461(02)80002-8
[29] Atul Gupta and Pankaj Jalote. 2007. An experimental evaluation of the effectiveness and efficiency of the test driven development. In *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 285–294. DOI: https://doi.org/10.1109/ESEM.2007.41
[30] Liang Huang and Mike Holcombe. 2009. Empirical investigation towards the effectiveness of test first programming. *Information and Software Technology* 51, 1 (2009), 182–194. DOI: https://doi.org/10.1016/j.infsof.2008.03.007
[31] ISO. 1994. *ISO 5725:1994. Accuracy (Trueness and Precision) of Measurement Methods and Results – Parts 1 to 6*. ISO.
[32] ISO/IEC. 2001. *ISO/IEC 9126-1:2001. Information Technology - Software Product Quality - Part 1: Quality Model*. ISO/IEC.

[33] ISO/IEC. 2011. *ISO/IEC 25010:2011. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models*. ISO/IEC.

[34] Joint Committee for Guides in Metrology. 2012. *International Vocabulary of Metrology – Basic and General Concepts and Associated Terms*. Number JCGM 200:2012 in 3. International Organization of Legal Metrology, Paris, France.

[35] Natalia Juristo. 2016. Experiences Conducting Experiments in Industry: The ESEIL FiDiPro Project. In *Proceedings of the 4th International Workshop on Conducting Empirical Studies in Industry (CESI '16)*. ACM, New York, NY, 1–3. DOI: https://doi.org/10.1145/2896839.2896846

[36] Barbara Kitchenham and Stuart Charters. 2007. Procedures for performing systematic literature review in software engineering. In *EBSE Technical Report Version 2.3, EBSE-2007-01*. Software Eng. Group., UK.

[37] Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. 1995. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering* 21, 12 (1995), 929–944. DOI: https://doi.org/10.1109/32.489070

[38] Barbara A. Kitchenham, Robert T. Hughes, and Stephen Linkman. 2001. Modeling software measurement data. *IEEE Transactions on Software Engineering* 27, 9 (2001), 788–804. DOI: https://doi.org/10.1109/32.950316

[39] Barbara A. Kitchenham, Shari L. Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering* 28, 8 (2002), 721–734. DOI: https://doi.org/10.1109/TSE.2002.1027796

[40] Lech Madeyski. 2005. Preliminary analysis of the effects of pair programming and test-driven development on the external code quality. In *Proceedings of the Conference on Software Engineering: Evolution and Emerging Technologies*. IOS Press, NLD, 113–123.

[41] Bertil Magnusson. 2003. *Handbook for Calculation of Measurement Uncertainty in Environmental Laboratories*. Nordtest.

[42] Bertil Magnusson, Håvard Hovind, Mikael Krysell, and T Naykki. 2007. Nordest handbook for calculation of measurement uncertainty based on quality control and method validation. In *Proceedings of 1st International Proficiency Testing Conference*. Springer, 26–30.

[43] Robert C. Martin. 2001. *Advanced Principles, Patterns and Process of Software Development*. Prentice Hall, UK.

[44] Matthias M Muller and O. Hagner. 2002. Experiment about test-first programming. *IEE Proceedings Software* 149, 5 (2002), 131–136. DOI: https://doi.org/10.1049/ip-sen:20020540

[45] Hussan Munir, Misagh Moayyed, and Kai Petersen. 2014a. Considering rigor and relevance when evaluating test driven development: A systematic review. *Information and Software Technology* 56, 4 (2014), 375–394. DOI: https://doi.org/10.1016/j.infsof.2014.01.002

[46] Hussan Munir, Krzysztof Wnuk, Kai Petersen, and Misagh Moayyed. 2014b. An experimental evaluation of test driven development vs. test-last development with industry professionals. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*. ACM, New York, NY, Article 50, 10 pages. DOI: https://doi.org/10.1145/2601248.2601267

[47] Glenford J. Myers, Corey Sandler, and Tom Badgett. 2012. *The Art of Software Testing* (3rd ed ed.). John Wiley & Sons, Hoboken and NJ.

[48] Nachiappan Nagappan, E. Michael Maximilien, Thirumalesh Bhat, and Laurie Williams. 2008. Realizing quality improvement through test driven development: Results and experiences of four industrial teams. *Empirical Software Engineering* 13, 3 (2008), 289–302. DOI: https://doi.org/10.1007/s10664-008-9062-z

[49] Teemu Näykki, Atte Virtanen, and Ivo Leito. 2012. Software support for the Nordtest method of measurement uncertainty evaluation. *Accreditation and Quality Assurance* 17, 6 (2012), 603–612.

[50] Lawrence W Neuman. 2002. *Social Research Methods: Qualitative and Quantitative Approaches*. Pearson Education Limited, Boston, USA.

[51] Andrea Padoan, Giorgia Antonelli, Ada Aita, Laura Sciacovelli, and Mario Plebani. 2017. An approach for estimating measurement uncertainty in medical laboratories using data from long-term quality control and external quality assessment schemes. *Clinical Chemistry and Laboratory Medicine* 55, 11 (2017), 1696–1701. DOI: https://doi.org/10.1515/cclm-2016-0896

[52] Matjaz Pancur and Mojca Ciglaric. 2011. Impact of test-driven development on productivity, code and tests: A controlled experiment. *Information and Software Technology* 53, 6 (2011), 557–573. DOI: https://doi.org/10.1016/j.infsof.2011.02.002

[53] Matjaz Pancur, Mojca Ciglaric, M. Trampus, and T. Vidmar. 2003. Towards empirical evaluation of test-driven development in a university environment. In *Proceedings of the IEEE Region 8 EUROCON 2003. Computer as a Tool*, Vol. 2. IEEE, 83–86. DOI: https://doi.org/10.1109/EURCON.2003.1248153

[54] Bartosz Papis, Konrad Grochowski, Kamil Subzda, and Kamil Sijko. 2022. Experimental evaluation of test-driven development with interns working on a real industrial project. *IEEE Transactions on Software Engineering* 48, 5 (2022), 1644–1664. DOI: https://doi.org/10.1109/TSE.2020.3027522

[55] Yahya Rafique and Vojislav B Misic. 2013. The effects of test-driven development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering* 39, 6 (2013), 835–856. DOI: https://doi.org/10.1109/TSE.2012.28

[56] Adrian Santos, Jaroslav Spisak, Markku Oivo, and Natalia Juristo. 2018. Improving development practices through experimentation: An industrial TDD case. In *Proceedings of the 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE Computer Society, 465–473. DOI: https://doi.org/10.1109/APSEC.2018.00061

[57] Adrian Santos, Sira Vegas, Oscar Dieste, Fernando Uyaguari, Ayşe Tosun, Davide Fucci, Burak Turhan, Giuseppe Scanniello, Simone Romano, Itir Karac, Marco Kuhrmann, Vladimir Mandic, Robert Ramac, Dietmar Pfahl, Christian Engblom, Jarno Kyykka, Kerli Rungi, Carolina Palomeque, Jaroslav Spisak, Markku Oivo, and Natalia Juristo. 2021. A family of experiments on test-driven development. *Empirical Software Engineering* 26 (2021), 1–53. DOI: https://doi.org/10.1007/s10664-020-09895-8

[58] Adrian Santos, Sira Vegas, Fernando Uyaguari, Oscar Dieste, Burak Turhan, and Natalia Juristo. 2020. Increasing validity through replication: An illustrative TDD case. *Software Quality Journal* 28, 2 (2020), 371–395. DOI: https://doi.org/10.1007/s11219-020-09512-3

[59] Dora Elizabeth Jaimes Sequeda. 2015. ?Se producen diferencias considerables en los resultados experimentales en función del procedimiento de medición? Un estudio comparativo para los constructos calidad y productividad en el marco de un experimento de test-driven development. Retrieved from https://oa.upm.es/44268/

[60] William Shadish, Thomas D. Cook, and Donald Thomas Campbell. 2002. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin Boston, MA, Boston, MA, USA.

[61] Brett Smith and Andrew C. Sparkes. 2016. *Routledge Handbook of Qualitative Research in Sport and Exercise*. Routledge, London, UK.

[62] Fiona Spring, Peter Gibbons, and Philip Tehan. 2001. Intra-examiner and inter-examiner reliability of a positional diagnostic screen for the lumbar spine. *Journal of Osteopathic Medicine* 4, 2 (2001), 47–55. DOI: https://doi.org/10.1016/S1443-8461(01)80002-2

[63] Matthew Thompson, Arpita Tiwari, Rongwei Fu, Esther Moe, and David I. Buckley. 2012. A framework to facilitate the use of systematic reviews and meta-analyses in the design of primary research studies. Retrieved from http://europepmc.org/books/NBK83621

[64] Zazie Todd, Nerlich, Brigitte, Suzanne Mckeown, and David D. Clarke. 2004. *Mixing Methods in Psychology: The Integration of Qualitative and Quantitative Methods in Theory and Practice*. Psychology Press, London, UK.

[65] Ayse Tosun, Oscar Dieste, Davide Fucci, Sira Vegas, Burak Turhan, Hakan Erdogmus, Adrian Santos, Markku Oivo, Kimmo Toro, Janne Jarvinen, and Natalia Juristo. 2017. An industry experiment on the effects of test-driven development on external quality and productivity. *Empirical Software Engineering* 22, 6 (2017), 2763–2805. DOI: https://doi.org/10.1007/s10664-016-9490-0

[66] Ayse Tosun, Oscar Dieste, Sira Vegas, Dietmar Pfahl, Kerli Rungi, and Natalia Juristo. 2021. Investigating the impact of development task on external quality in test-driven development: An industry experiment. *IEEE Transactions on Software Engineering* 47, 11 (2021), 2438–2456. DOI: https://doi.org/10.1109/TSE.2019.2949811

[67] John V. Vu, Niklas Frojd, Clay Shenkel-Therolf, and David S. Janzen. 2009. Evaluating test-driven development in an industry-sponsored capstone project. In *Proceedings of the 6th International Conference on Information Technology: New Generations*. IEEE, 229–234. DOI: https://doi.org/10.1109/ITNG.2009.11

[68] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer, Berlin.

Received 5 December 2023; revised 2 June 2024; accepted 12 July 2024