# Which demographic variables better characterize a population?

Oscar Dieste
*Universidad Politécnica de Madrid*
Madrid, Spain
odieste@fi.upm.es

Silvia T. Acuña
*Universidad Autónoma de Madrid*
Madrid, Spain
silvia.acunna@uam.es

Natalia Juristo
*Universidad Politécnica de Madrid*
Madrid, Spain
natalia@fi.upm.es

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

## I. Introduction

Los factores humanos influyen en el proceso de desarrollo de software. Las comunidades científica y profesional generalmente asumen que factores como la experiencia o el conocimiento tienen efectos positivos sobre la X [REFS], Y [REFS] o Z [REFS].

No obstante, existen resultados negativos. [NOTA OSCAR: Ir al paper y quizás a la tesis de Alejandrina, e indicar casos de resultados negativos]

Nuestro objetivo es identificar qué factores humanos son relevantes.

Poner la sección 3 aquí.

## II. Background

### A. Related works

It is unclear which personal aspects influence TDD programming. The most comprehensive review has been performed by Raura *et al.* [] using both primary and secondary studies on TDD. The authors found that the *overall professional experience*, *TDD experience*, *programming language experience*, and *training* were the personal aspects most frequently examined in the empirical research. Evidence is somewhat conflicting, but these personal aspects seem to influence TDD performance positively. A remarkable exception happens when the experience is dichotomized into the *professional* and *student* categories: Professionals seem less productive than students.

During the FiDiPro ESEIL project [1], we conducted multiple TDD experiments in Industry and Academy[1]. We collected the participants' demographic information using a survey[2]. The surveys evolved during the project, but they all inquired about the participants' education, training, experience, and knowledge from different perspectives, e.g., testing, methodologies, programming languages, etc. Due to the small sample size at the experiment level, we did not individually analyze the influence of the personal aspects on TDD performance (except for Fucci *et al.* [2], whose results were inconclusive).

At the project's end, Dieste *et al.* [3] performed a joint analysis.[3] As far as we know, this is the most extensive study of the impact of personal aspects on TDD performed to date. The analyses yielded a mixture of intuitive and counter-intuitive results:

1) **Intuitive**: When practicing ITL programming, professionals achieve higher quality and productivity than students. Likewise, participants with a degree in Computing exhibit better quality and productivity[4].
2) **Counter-intuitive**: When practicing TDD, professionals achieve lower quality and productivity than students. Likewise, participants with a degree in Computing exhibit worse quality and productivity. These results contradict common sense[5], although, as indicated above, *literature reviews on TDD have reported that professionals seem less productive than students*.
3) **Intuitive**: The programming experience acquired by the participants during their formal education improves quality and productivity, regardless the participants are using ITL programming or TDD[6].
4) **Counter-intuitive**: The programming experience acquired by the participants during their formal education is the only experience that influences ITL programming or TDD. However, according to the literature and common sense, *other experiences* should have an effect too, e.g., programming experience acquired in industry.

[1]The experiments conducted during the ESEIL project are described [NOTA OSCAR: Poner una URL correcta] here.

[2]All surveys are available from the URL specified in footnote #1. The raw answers are also available in the same URL.

[3]Due to the characteristics of the special issue where the paper was published, Dieste *et al.* [3] only reports the influence of personal factors on ITL (Iterative Test-Last) programming. We report in this paper the influence of personal factors on TDD (see Appendix B). Readers can perform such analyses using the dataset and the SPSS© analysis scripts reported in [3].

[4]The concepts of quality and productivity are intuitive. For a formal definition, see [3].

[5]Please check the statistical significances in Appendix B. Several variables are non-significant, but the trends are clear.

[6]Statistical significance was not reached for quality when participants performed TDD; see Appendix B.

### B. *Possible explanations*

Counter-intuitive results should have an explanation. For the item #2 above, we believe the results are artifactual. Dieste *et al.* [4] reports that age is related to a particular type of drop-off characterized by not finalizing the experimental tasks. Senior participants work in the experimental tasks at a slower pace, causing the impression of diminished performance. Ultimately, it is a motivation issue.

The reason for the item #4 is less obvious. If the academic programming experience has an effect, the industry programming experience should have a (positive) effect as well. Furthermore, the effect of academic programming experience holds both for ITL and TDD, so it does not look like an artifactual effect, i.e., unrelated to senior participants' motivation. In our opinion, two explanations are possible:

- **Experience only sometimes leads to better performance**: McDaniel *et al.* [5] reported low correlations between experience and performance. Camerer and Johnson [6] conclude that subjects with experience make decisions or predictions that are no better (or even worse) than those made by inexperienced subjects. There are studies with similar outcomes in programming [7], [8], as well as in other areas of SE, e.g., [9], [10]. This apparent contradiction can be explained if a distinction is made between experience and expertise. To achieve expert performance, subjects need to complete a period of intensive practice with the deliberate intention of improving performance (i.e., acquiring expertise). The mere exercise of an activity (i.e., the industry programming experience) may improve performance but not to the point of it being equal to that of people generally recognized as experts in an area [11].
- **Measurement problem**: Academic programming experience is relatively well-defined. Students do lots of programming in their degrees. However, industry programming experience could be more challenging to estimate. In a regular workday, practitioners meet, answer emails, write reports, and sometimes code. On top of that, different professionals spend different times on each of these activities, and the times evolve over the practitioners' careers.

The first explanation has been explored in Dieste *et al.* [3]. The authors could not find evidence against it, i.e., such an explanation is consistent with the empirical data. This paper will explore the second explanation, i.e., whether the lack of relationship and superior performance may be due to measurement problems.

## III. Methodology

https://stats.stackexchange.com/questions/215404/is-there-factor-analysis-or-pca-for-ordinal-or-binary-data

https://www.youtube.com/watch?v=crbZa5XRVh4

### A. *Dataset(s)*

We collected information about 178 experimental subjects (84 professionals and 94 students). A curated version of the dataset is available at http://www.grise.upm.es/sites/extras/11/.

### B. *Measurement problems*

In some cases, e.g., the different types of experience, we requested the same information using different scales (*Likert* and *number of years*) to avoid the risk of bias in the responses [12].

### C. *Research questions*

More concretely, this paper aims to answer the following research questions:

**RQ1.** Is it possible to summarize the data, e.g., pooling together different types of experience so that a "more solid definition" of experience shows up? In other words, if we apply a dimension-reduction procedure to the demographic data, does such a procedure yield interpretable results?

**RQ2.** Does the dimension-reduced demographic data predict the code quality and programmers' productivity more accurately than the original, non-reduced demographic data?

### D. *Analysis method*

We will use the Exploratory Factor Analysis (EFA) for dimensionality reduction. The reason is twofold: 1) EFA has been often used in software engineering [13], [14], and 2) EFA's output can be interpreted as non-observable *constructs* that underlie the experimental data [15, Chapter 1]. The analyses have been conducted with *R* [16].

In the realm of Psychology, a construct is *a label for a cluster or domain of covarying behaviours* [17]. For instance, intelligent people are expected to perform well on math or language tests. Math and language test scores are "covarying behaviours" because they tend to increase/decrease together depending on whether the people are more/less intelligent. Intelligence plays, in this case, the role of construct.

Outside Psychology, the concept of construct receives other names, such as *latent variable* in Structural Equation Modelling [18]. We will use the term "construct" because Bacharach's theory components [19] are frequently used and well-known in software engineering, e.g., [20], [21].

EFA is frequently used in social sciences to explore the psychometric properties of a survey [22]. Surveys contain several related questions because it is challenging for a single question to provide the sought information accurately. The EFA helps to identify which queries are associated with the same construct. This is precisely the starting point for this research.

### E. *Reproducibility*

The demographic data is available online (see Section III-A). The paper in `.Rtex` mode (including embedded R code) is be available at ADDGITHUB.

## IV. Results

We will follow Watkins' guidance [15, pp. 33] to conduct the analysis. It is the best book, in our humble opinion, about EFA.

The variables in our dataset are binary and ordinal; we will apply polychoric correlations instead of Pearson correlations accordingly.

## A. *Participants and variables*

The demographic variables are listed in Appendix A.

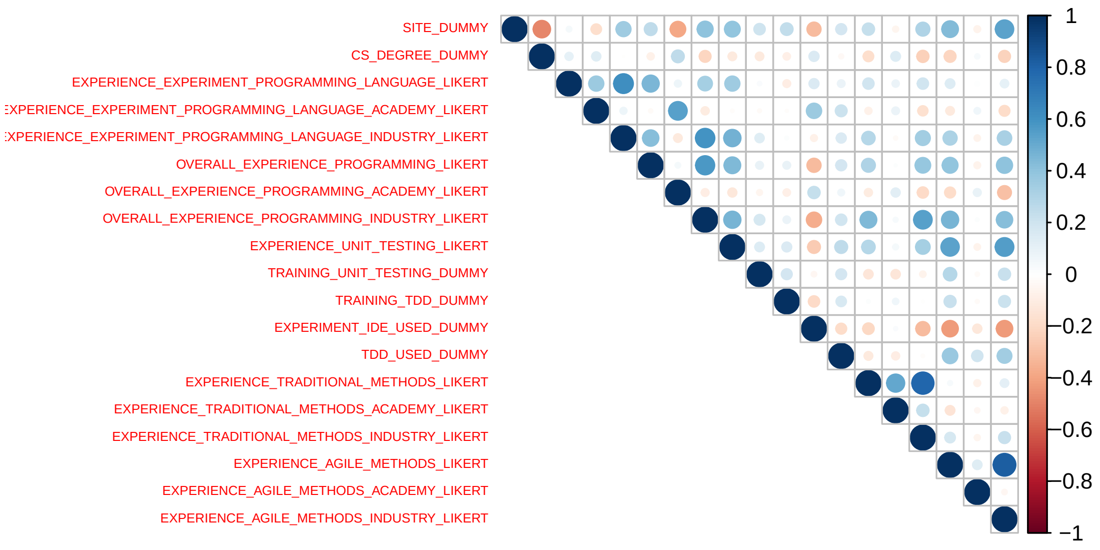
## B. *Factorability considerations*

Figure 1 shows the biseral correlations

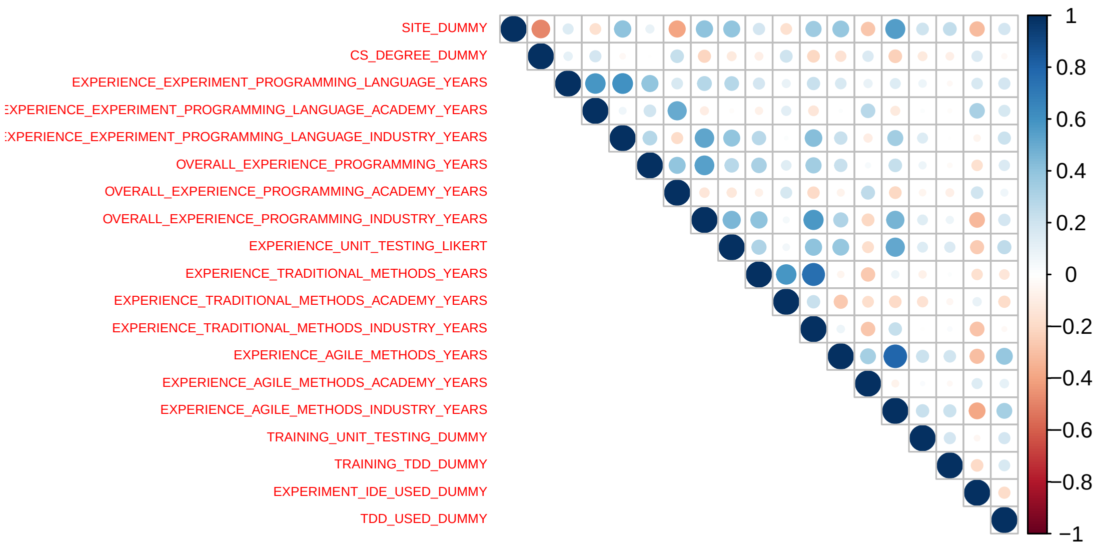
As a side effect, we obtained somewhat redundant information. For instance, question #4 about *overall programming experience* subsumes, to some extent, question #3 regarding the programming languages used by the subject. It was not a surprise that the data were highly correlated, as figures 1a and 1b show.

The variables measured in Likert and year scales are also correlated. However, these correlations are not interesting to this research (please note that Figure 1a includes variables measured in a Likert scale and Figure 1b variables measured in years). We collected both scales because our previous research showed that students' responses are biased when subjective scales, e.g., a five-point Likert scale (1=low and 5=high), are used [12]. Other researchers have recently come to similar conclusions [23] (but others not, e.g., [13]).

contar aqui lo de los niveles detallado y simplificado



(a) Measured using Likert scales



(b) Measured using years

Fig. 1: Biserial (Kendall) correlations between demographic variables. We are not using Pearson correlations because some variables, e.g., CS_DEGREE, are dummies or have been measured in a Likert scale, e.g., EXPERIENCE_UNIT_TESTING.

There are several, sometimes conflicting recommendations to assess the factorability of a dataset. Three criteria frequently mentioned in textbooks are:

- There are several correlations with values $r > 0.30$. This fact can be easily assessed by looking at Figure 1. There are more and higher correlations when the demographic variables are measured in years (Figure 1b). Intuitively, that dataset is "better", and the final results corroborate this impression.
- The determinants of the correlation matrices are 0.0002 and 0.0001). A dataset is not affected by multicollinearity when the determinat $> 0.00001$ [15, Chapter 9].
- The variable/sample size ratio exceeds $1 : 5$, although some authors recommend higer values ($1 : 10$ or even $1 : 20$) [15]. The four datasets contain 124 cases, yielding $1 : 11.3$, $1 : 11.3$, $1 : 8.3$, and $1 : 8.3$ ratios, respectively.
- The Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy estimates the proportion of variance that can be assigned to the underlying constructs [15]. EFA can be applied to a dataset when $KMO \geq 0.6$ [24]. The KMO values for our datasets are 0.7, 0.65, 0.7, and 0.7, respectively.
- The Bartlett's test of sphericity checks whether the variables in the original dataset are redundant. The null hypothesis states that the variables are not redundant. Dimension reduction is thus possible when Bartlett's test achieves statistical significance (typically at the $\alpha = 0.05$ level) [15]. In the four datasets, p-value $< 0.001$.

## C. Number of factors

The visual examination of the scree plot usually defines the number of factors to extract. However, we have used an alternate approach provided by the `fa.parallel()` function from the `psych` package [25]. This function uses the "parallel" analysis proposed by Horn [26]. The advantage of the parallel approach vs. scree plot examination is objectivity. Nevertheless, other "objective" approaches exist (see, e.g., the help page of the `fa.parallel()` function or [15, Chapter 12]).

Datasets 1 – 3 can be summarized using three factors. In the case of the dataset 4, the function `fa.parallel()` suggests four factors[7].

## D. Factor extraction

The factors have been extracted using the `fa` function [25]. We have used Ordinary Least Squares as extraction method because it works well with few data points and non-normal data [15, Chapter 11]. We have used an oblique rotation (Promax) because it is unlikely, from the outset, that the associated constructs are uncorrelated, e.g., different types of experiences depend, to a large extent, on the passing of time.

There are two types of factor loadings when oblique rotations are applied: *pattern* and *structure*. They represent different quantitative measures of the relationship between the factors and the demographic variables, and both should be interpreted [15, Chapter 13]. In our case, the pattern and structure matrices are similar, so for brevity, we will interpret only the former.

## E. Which analysis looks better?

The pattern factor loadings are shown in Tables II and II for the simplified and detailed datasets, respectively. There are no significant differences among the tables, i.e., they extract quite similar factors in all four cases (we have rearranged the output of the `fa` function to emphasize the similarities). However, as discussed in Section II, we have collected demographic variables with different scales and abstraction levels, so it is possible that a given dataset yields better results than others. There are three criteria to evaluate the factors extracted by the EFA:

1) The factor loadings should be high, i.e., the underlying constructs strongly relate to the demographic variables.
2) The factors should load on different demographic variables, i.e., constructs are independent.
3) All demographic variables load on a factor, i.e., no unidentified constructs exist.

These criteria cannot be achieved to some degree only. The trade-offs are somewhat subjective, and thus the final conclusion. From our perspective:

1) The loadings are slightly higher when the demographic variables are measured in years.
2) The number of overlaps is smaller for the simplified datasets.
3) The variable EXPERIMENT_IDE_USED_DUMMY loads when only the demographic variables are measured in Likert scales. TRAINING_TDD_DUMMY never loads, but it makes complete sense because most of the people never received such training, e.i., most values are zero.

Considering the pros and cons, we choose Table I (left) as the best set of factors (simplified dataset, measured in the Likert scale). This will be particularly relevant when we discuss RQ2. However, all tables look very similar, and the relevant underlying constructs are almost the same in all cases, as described in the next section.

## V. Discussion

### A. RQ1: Construct identification

We have assigned construct names to all the factors in Tables I and II. The similarities are apparent. We will use Table I (left) to illustrate the naming approach:

- The first factor loads on:
  - EXPERIENCE EXPERIMENT PROGRAMMING LANGUAGE
  - OVERALL EXPERIENCE PROGRAMMING
  - EXPERIENCE UNIT TESTING
  - EXPERIENCE TRADITIONAL METHODS

[7]Due to the randomness that underlies `fa.parallel()` (the comparison between the observed data and a random data matrix), in some executions, the suggested number of factors is three.

TABLE I: Factor loadings for the simplified dataset. The Likert-type variables are displayed on the left. The year-type variables are on the right. The constructs have been assigned a name for a more straightforward interpretation

| Variable (Likert) | Prog. experience | Age | Method. practice | Prog. experience | Age | Method. practice | Variable (Years) |
|---|---|---|---|---|---|---|---|
| SITE_DUMMY | | 0.71 | | | 0.85 | | SITE_DUMMY |
| CS_DEGREE_DUMMY | | -0.58 | | | -0.54 | | CS_DEGREE_DUMMY |
| EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_LIKERT | 0.92 | -0.31 | | 0.71 | | | EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_YEARS |
| OVERALL_EXPERIENCE_PROGRAMMING_LIKERT | 0.6 | | | 0.73 | | | OVERALL_EXPERIENCE_PROGRAMMING_YEARS |
| EXPERIENCE_UNIT_TESTING_LIKERT | 0.56 | | 0.32 | 0.32 | | 0.41 | EXPERIENCE_UNIT_TESTING_LIKERT |
| EXPERIENCE_TRADITIONAL_METHODS_LIKERT | 0.37 | 0.45 | -0.45 | 0.73 | | -0.31 | EXPERIENCE_TRADITIONAL_METHODS_YEARS |
| EXPERIENCE_AGILE_METHODS_LIKERT | | | 0.69 | | | 0.85 | EXPERIENCE_AGILE_METHODS_YEARS |
| TRAINING_UNIT_TESTING_DUMMY | | | 0.35 | | | | TRAINING_UNIT_TESTING_DUMMY |
| TRAINING_TDD_DUMMY | | | | | | | TRAINING_TDD_DUMMY |
| EXPERIMENT_IDE_USED_DUMMY | | -0.47 | | | -0.3 | | EXPERIMENT_IDE_USED_DUMMY |
| TDD_USED_DUMMY | | | 0.52 | | | 0.62 | TDD_USED_DUMMY |

TABLE II: Factor loadings for the simplified dataset. The table is organized the same as Table I

| Variable (Likert) | Prog. experience | Age | Method. practice | Prog. experience | factor 4? | Method. practice | Age? | Variable (Years) |
|---|---|---|---|---|---|---|---|---|
| SITE_DUMMY | | -0.31 | 0.4 | | | 0.33 | -0.58 | SITE_DUMMY |
| CS_DEGREE_DUMMY | | 0.3 | | | | | 0.68 | CS_DEGREE_DUMMY |
| EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_ACADEMY_LIKERT | | 0.75 | | | 0.87 | | | EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_ACADEMY_YEARS |
| EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_INDUSTRY_LIKERT | 0.67 | | | 0.67 | | | | EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_INDUSTRY_YEARS |
| OVERALL_EXPERIENCE_PROGRAMMING_ACADEMY_LIKERT | | 0.77 | | | 0.56 | | | OVERALL_EXPERIENCE_PROGRAMMING_ACADEMY_YEARS |
| OVERALL_EXPERIENCE_PROGRAMMING_INDUSTRY_LIKERT | 0.81 | | | 0.73 | 0.32 | | | OVERALL_EXPERIENCE_PROGRAMMING_INDUSTRY_YEARS |
| EXPERIENCE_UNIT_TESTING_LIKERT | 0.47 | | 0.41 | 0.51 | | 0.5 | | EXPERIENCE_UNIT_TESTING_LIKERT |
| EXPERIENCE_TRADITIONAL_METHODS_ACADEMY_LIKERT | 0.31 | | | | | -0.3 | | EXPERIENCE_TRADITIONAL_METHODS_ACADEMY_YEARS |
| EXPERIENCE_TRADITIONAL_METHODS_INDUSTRY_LIKERT | 0.91 | | -0.35 | 0.92 | | | | EXPERIENCE_TRADITIONAL_METHODS_INDUSTRY_YEARS |
| EXPERIENCE_AGILE_METHODS_ACADEMY_LIKERT | | | | | | | | EXPERIENCE_AGILE_METHODS_ACADEMY_YEARS |
| EXPERIENCE_AGILE_METHODS_INDUSTRY_LIKERT | | | 0.7 | 0.3 | | 0.83 | | EXPERIENCE_AGILE_METHODS_INDUSTRY_YEARS |
| TRAINING_UNIT_TESTING_DUMMY | | | 0.37 | | | | | TRAINING_UNIT_TESTING_DUMMY |
| TRAINING_TDD_DUMMY | | | | | | | | TRAINING_TDD_DUMMY |
| EXPERIMENT_IDE_USED_DUMMY | | 0.31 | | | | | | EXPERIMENT_IDE_USED_DUMMY |
| TDD_USED_DUMMY | | | 0.62 | | | 0.58 | | TDD_USED_DUMMY |

A construct can be understood as a continuous variable in the EFA framework. Small values of this construct correspond to people with small values of the related demographic variables and vice versa. The three first variables refer to programming abilities; thus, the construct could be interpreted as "Programming experience." The fourth variable (EXPERIENCE TRADITIONAL METHODS) is not related to programming, but experience requires time, and people with high experience probably have used traditional methods in the past.

- The second factor loads on:
  - SITE, a dummy variable representing the site where the experiment was conducted: Academia(0), Industry (1).
  - CS DEGREE, a dummy variable representing whether the subject did not have a degree in CS (0) or does have it (1). Please notice that the loading, in this case, is negative.
  - EXPERIENCE EXPERIMENT PROGRAMMING LANGUAGE. The loading is negative again.
  - EXPERIENCE TRADITIONAL METHODS
  - EXPERIMENT IDE USED. The loading is negative again.

  So, this construct has, on one side, experiments conducted in Industry with people without a CS degree. They have experience with traditional methods but not with Eclipse. They do not know Java very well. On the other, experiments conducted in academia with people with CS degrees, Java, and Eclipse knowledge but little knowledge of traditional methods. The most likely interpretation for this construct is "Professionalism" (practitioner vs. student), but we are not convinced. There are many professionals with a CS degree nowadays, and the knowledge of traditional methods suggests that the subjects have worked for quite a long time. Therefore, we venture out to name this construct "Age."
- Applying a similar procedure, we believe that the third factor could be named "Methodological practice," having in the low values those subjects who have used traditional methods and in the high values those who have used agile methods.

The fourth factor in Table II (right) cannot be interpreted because it loads on too few demographic variables.

### B. RQ2: Construct identification

## VI. Limitations

Many limitations exist: It is a limited dataset, obtained in just a few countries and within a concrete family of experiments, etc. Nevertheless, these issues are common in empirical research. In this section, we wish to point out more specific limitations:

- The best analysis corresponds to dataset 1, where the demographic variables are measured in Likert scales. We expected that the variables measured in years would be more informative. A likely reason is that several variables measured in years were obtained using open questions that required manual post-processing and could introduce noise.
- The names of the constructs are subjective interpretations of the factor loadings. For instance, the "Programming experience" construct loads on EXPERIENCE UNIT TESTING. Unit testing, mainly if it is automated, suggests that the programming experience is relatively recent and enters in contradition with the interpretation given to the variable EXPERIENCE TRADITIONAL METHODS. More data would be necessary for a more accurate analysis, but the EFA provides at least some working hypotheses.
-

## VII. Conclusions

Responder a las preguntas de investigación

## VIII. Acknowledgements

This work was supported by the projects PGC2018-097265-B-I00 and RTI2018-095255-B-I00 (MASSIVE project) funded by the Spanish Ministry of Science and Innovation and co-financed by FEDER, and the Madrid Region R&D program (project FORTE, P2018/TCS-4314).

## References

[1] N. Juristo, "Experiences conducting experiments in industry: the eseil fidipro project," in *Proceedings of the 4th International Workshop on Conducting Empirical Studies in Industry*, 2016, pp. 1–3.

[2] D. Fucci, B. Turhan, N. Juristo, O. Dieste, A. Tosun-Misirli, and M. Oivo, "Towards an operationalization of test-driven development skills: An industrial empirical study," *Information and Software Technology*, vol. 68, pp. 82–97, 2015.

[3] O. Dieste, A. Aranda, F. Uyaguari, B. Turhan, A. Tosun, D. Fucci, M. Oivo, and N. Juristo, "Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018, pp. 111–112.

[4] O. Dieste, G. Raura, P. Rodríguez *et al.*, "Professionals are not superman: Failures beyond motivation in software experiments," in *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)*. IEEE, 2017, pp. 27–32.

[5] M. A. McDaniel, F. L. Schmidt, and J. E. Hunter, "Job experience correlates of job performance." *Journal of applied psychology*, vol. 73, no. 2, p. 327, 1988.

[6] C. F. Camerer and E. J. Johnson, "The process-performance paradox in expert judgment-how can experts know so much and predict so badly?" 1991.

[7] M. M. Muller and F. Padberg, "An empirical study about the feelgood factor in pair programming," in *10th International Symposium on Software Metrics, 2004. Proceedings.* IEEE, 2004, pp. 151–158.

[8] S. B. Sheppard, B. Curtis, P. Milliman, and T. Love, "Modern coding practices and programmer performance," *Computer*, vol. 12, no. 12, pp. 41–49, 1979.

[9] G. M. Marakas and J. J. Elam, "Semantic structuring in analyst acquisition and representation of facts in requirements analysis," *Information Systems Research*, vol. 9, no. 1, pp. 37–63, 1998.

[10] S. Sonnentag, "Excellent software professionals: Experience, work activities, and perception by peers," *Behaviour & Information Technology*, vol. 14, no. 5, pp. 289–299, 1995.

[11] K. A. Ericsson and N. Charness, "Expert performance: Its structure and acquisition." *American psychologist*, vol. 49, no. 8, p. 725, 1994.

[12] A. Aranda, O. Dieste, and N. Juristo, "Evidence of the presence of bias in subjective metrics: Analysis within a family of experiments," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–4.

[13] J. Siegmund, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg, "Measuring and modeling programming experience," *Empirical Software Engineering*, vol. 19, no. 5, pp. 1299–1334, 2014.

[14] A. Barcomb, K.-J. Stol, D. Riehle, and B. Fitzgerald, "Why do episodic volunteers stay in floss communities?" in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 948–959.

[15] M. Watkins, *A Step-by-Step Guide to Exploratory Factor Analysis with R and RStudio*. Taylor & Francis, 2020.

[16] R Core Team, *R: Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022. [Online]. Available: https://www.R-project.org/

[17] "construct – psychology," https://www.britannica.com/science/construct, [Accessed 01-Oct-2022].

[18] R. Kline, *Principles and Practice of Structural Equation Modeling, Fourth Edition*, ser. Methodology in the Social Sciences. Guilford Publications, 2015.

[19] S. B. Bacharach, "Organizational theories: Some criteria for evaluation," *Academy of management review*, vol. 14, no. 4, pp. 496–515, 1989.

[20] N. Juristo and A. Moreno, *Basics of Software Engineering Experimentation*. Springer US, 2013.

[21] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, ser. Computer Science. Springer Berlin Heidelberg, 2012.

[22] A. B. Costello and J. Osborne, "Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis," *Practical Assessment, Research, and Evaluation*, vol. 10, no. 1, p. 7, 2005.

[23] M. Daun, J. Brings, P. A. Obe, and V. Stenkova, "Reliability of self-rated experience and confidence as predictors for students' performance in software engineering," *Empirical Software Engineering*, vol. 26, no. 4, pp. 1–41, 2021.

[24] H. F. Kaiser and J. Rice, "Little jiffy, mark iv," *Educational and psychological measurement*, vol. 34, no. 1, pp. 111–117, 1974.

[25] W. Revelle, *psych: Procedures for Psychological, Psychometric, and Personality Research*, Northwestern University, Evanston, Illinois, 2022, r package version 2.2.9. [Online]. Available: https://CRAN.R-project.org/package=psych

[26] J. L. Horn, "A rationale and test for the number of factors in factor analysis," *Psychometrika*, vol. 30, no. 2, pp. 179–185, 1965.

## APPENDIX

### A. *List of demographic variables*

[NOTA OSCAR: Describir las variables con un breve texto.]

#### *1) Dichotomous variables:*

- SITE_DUMMY
- CS_DEGREE_DUMMY
- TRAINING_UNIT_TESTING_DUMMY
- TRAINING_TDD_DUMMY
- EXPERIMENT_IDE_USED_DUMMY
- TDD_USED_DUMMY

#### *2) Experiences measured in a Likert scale:*

- EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_ACADEMY_LIKERT
- EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_INDUSTRY_LIKERT
- OVERALL_EXPERIENCE_PROGRAMMING_ACADEMY_LIKERT
- OVERALL_EXPERIENCE_PROGRAMMING_INDUSTRY_LIKERT
- EXPERIENCE_UNIT_TESTING_LIKERT
- EXPERIENCE_TRADITIONAL_METHODS_ACADEMY_LIKERT
- EXPERIENCE_TRADITIONAL_METHODS_INDUSTRY_LIKERT
- EXPERIENCE_AGILE_METHODS_ACADEMY_LIKERT
- EXPERIENCE_AGILE_METHODS_INDUSTRY_LIKERT

#### *3) Experiences measured in years:*

- EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_ACADEMY_YEARS
- EXPERIENCE_EXPERIMENT_PROGRAMMING_LANGUAGE_INDUSTRY_YEARS
- OVERALL_EXPERIENCE_PROGRAMMING_ACADEMY_YEARS
- OVERALL_EXPERIENCE_PROGRAMMING_INDUSTRY_YEARS
- EXPERIENCE_TRADITIONAL_METHODS_ACADEMY_YEARS
- EXPERIENCE_TRADITIONAL_METHODS_INDUSTRY_YEARS
- EXPERIENCE_AGILE_METHODS_ACADEMY_YEARS
- EXPERIENCE_AGILE_METHODS_INDUSTRY_YEARS

## *B. Analyses not reported in Dieste et al. [3]*

TABLE III: Analysis of the influence of personal aspects on code quality using TDD. This analysis has not been reported in [3].

| **Model** | **Variables** | **Unstandardized Coefficients** | | **Standardized Coefficients** | **t** | **Sig.** |
|---|---|---|---|---|---|---|
| | | **B** | **Std. Error** | **Beta** | | |
| 1 | (Constant) | 68.980 | 19.696 | | 3.502 | 0.001 |
| | Environment (lab/industry) | -9.375 | 7.413 | -0.157 | -1.265 | 0.209 |
| | Have a degree in CS | -7.119 | 7.126 | -0.104 | -0.999 | 0.320 |
| | Experience in the unit testing framework used during the experiment | -10.633 | 6.517 | -0.198 | -1.632 | 0.106 |
| | Experience in the programming language used in the experiment acquired in academy | 1.024 | 1.382 | 0.093 | 0.741 | 0.460 |
| | Experience in the programming language used in the experiment acquired in industry | 0.625 | 1.421 | 0.065 | 0.440 | 0.661 |
| | Overall experience in programming acquired in academy | 1.048 | 0.932 | 0.126 | 1.124 | 0.263 |
| | Overall experience in programming acquired in industry | -0.969 | 0.795 | -0.195 | -1.219 | 0.226 |
| | Experience in unit testing | 7.794 | 5.175 | 0.191 | 1.506 | 0.135 |
| | Current usage of the IDE used in the experiment | -2.138 | 6.223 | -0.035 | -0.344 | 0.732 |
| | Current usage of TDD | 11.033 | 8.042 | 0.140 | 1.372 | 0.173 |
| | Task performed using TDD | 9.926 | 10.607 | 0.162 | 0.936 | 0.351 |
| | Was the TDD task presented in slices? | 12.680 | 10.608 | 0.208 | 1.195 | 0.235 |

*a. Dependent Variable: External quality achieved using TDD*

TABLE IV: Analysis of the influence of personal aspects on programmers' productivity using TDD. This analysis has not been reported in [3]. Significant results are highlighted in red.

| **Model** | **Variables** | **Unstandardized Coefficients** | | **Standardized Coefficients** | **t** | **Sig.** |
|---|---|---|---|---|---|---|
| | | **B** | **Std. Error** | **Beta** | | |
| 1 | (Constant) | 21.928 | 20.895 | | 1.049 | 0.296 |
| | Environment (lab/industry) | -16.668 | 7.864 | -0.251 | -2.120 | 0.036 |
| | Have a degree in CS? | -7.723 | 7.560 | -0.101 | -1.022 | 0.309 |
| | Experience in the unit testing framework used during the experiment | 5.550 | 6.914 | 0.093 | 0.803 | 0.424 |
| | Experience in the programming language used in the experiment acquired in academy | -1.331 | 1.466 | -0.109 | -0.907 | 0.366 |
| | Experience in the programming language used in the experiment acquired in industry | -0.882 | 1.508 | -0.082 | -0.585 | 0.560 |
| | Overall experience in programming acquired in academy | 2.692 | 0.989 | 0.291 | 2.723 | 0.008 |
| | Overall experience in programming acquired in industry | 0.279 | 0.844 | 0.050 | 0.331 | 0.741 |
| | Experience in unit testing | 8.934 | 5.490 | 0.196 | 1.627 | 0.107 |
| | Current usage of the IDE used in the experiment | 3.231 | 6.602 | 0.047 | 0.489 | 0.626 |
| | Current usage of TDD | -6.190 | 8.531 | -0.071 | -0.726 | 0.470 |
| | Task performed using TDD | 12.769 | 11.253 | 0.187 | 1.135 | 0.259 |
| | Was the TDD task presented in slices? | 14.985 | 11.254 | 0.220 | 1.331 | 0.186 |

*a. Dependent Variable: Productivity achieved using TDD*