A maturity model for Test-Driven Development adoption in professional settings

Matheus Marabesi[1][0000−0001−7646−554X], Alicia García-Holgado[2][0000−0001−9663−1103], and Francisco José García-Peñalvo[3][0000−0001−9987−5584]

GRIAL Research Group - Universidad de Salamanca (https://ror.org/02f40zc51), Patio de Escuelas, 1, Salamanca, 37008, Spain

Abstract. This paper presents preliminary results on developing a Test-Driven Development (TDD) maturity model, incorporating organizational context and testing strategy. A qualitative study provided initial insights, while an evaluation by experts study further refined and evaluated each level. Our results indicate that the model captures the complexity of TDD adoption in professional settings, suggesting its applicability for evaluating and improving TDD practices. Future work will refine the model and explore its application in different organizational contexts.

Keywords: TDD · Maturity Model · Testing Strategy · Unit testing · Test Smells.

1 Introduction

The idea of test-driven development (TDD) has been around since 1950's [5], however, its popularity gained traction later on with the work by Kent Beck [7] [8] and other authors [23] [3] that started to discuss the practice and contributed to the literature for adoption from practitioners.

According to Kent Beck TDD is a bit nebulous, TDD is an awareness of the gap between decision and feedback during the programming phase. Beck mentions the gap between feedback and the program, aiming at the difficulty for programmers in having a quick response at the moment they are writing the code. The classic writing code process by the person who develops software follows three steps. The first is writing the code, the second is testing and the third is changing the code in case something is wrong or there is a need to improve it. Note that the test happens after the code has been written.

TDD has been a discussed subject that presents different results. The studies over the years about TDD point to improvements in external quality [27] [24] [10] [9], and, for some studies, it points to inconclusive on internal quality [1], even the practice of TDD has been challenged by researchers [17]. When student settings are taken into account the results are mixed [31]. The last 13 years of systematic literature reviews depict this scenario in detail [22]. Practitioners use TDD to build software daily in different manners. Despite evidence of dropping the practice due to pressure [11].

However, the literature points to a lack of structure in a model that practitioners and researchers can build on top of it to establish a baseline and monitor the progress of its adoption. TDD has been closely related to code quality, and its usage has been related to the measure of code coverage [26]. In that sense, TDD as a practice has been lacking a model on how to adopt it effectively by organizations and teams of software development. This paper elaborate further on that, presenting the preliminary evaluation of the TDD maturity model that was created based on the CMMI model developed by the Software Engineering Institute at Carnegie Mellon University [12] that focuses on three key areas: Requirements Engineering, Technical Enabler and Team Settings.

The remain of this paper is structured as follows. Section 2 presents related work on TDD and maturity models. Section 3 elaborates on the why a maturity model for TDD. Section 4 presents the TDD maturity model. Section 5 elabores on the preliminary evaluation, Section 6 presents the evaluation of the model by experts. Section 7 presents the conclusion and future work.

2 Related work

The concept of maturity model was used by Richardson [28] to categorize the levels of maturity for RESTful APIs [14]. The model is linear and uses four levels. The maturity model guides the adoption of Restful services and shapes the ways of interacting with its consumers.

Researchers also tackled the challenge of understanding where organizations stand regarding DevOps. Kurkela [20] conducted a study to determine the state of DevOps within a team of 17 members. The initial analysis of the team's ways of working shows that the team uses Scrum as a framework to develop software, but the framework is not utilized to its full capacity. The framework itself does not have any formal recommendations regarding testing practices. Therefore, in the literature, Tarlinder [30] presented a perspective to fill in this gap. The author utilizes the maturity model developed by Zarour et al. [32], which consists of five levels of maturity.

TDD has been a subject that researchers focused on developing guidelines and the outcomes of the practice of observing developers' TDD flow. The work by Aniche and Gerosa surveyed 218 programmers and cataloged 9 of the most common mistakes in the TDD practice [4]. In the other spectrum, the work by Staegemann et al. [29] presented twenty guidelines based on a Structured Literature Review to offer practitioners on the path towards TDD adoption based on scientific production. The guidelines are cross-functional and go from learning to practices that might prevent long-term issues when practicing TDD.

Based on the related work presented, the following section elaborate on the context of TDD and conceptualize the foundation of a maturity model tailored for it.

3 Why a maturity model?

DevOps metrics marked a turning point for businesses to focus on technical practices known by teams that excelled at delivering software with the speed and precision required by the business [15]. These metrics emphasize outcomes rather than outputs, and they are as follows:

- Lead Time (LT): Measures the duration from when a request is made until it is available for customer use (often related to deployment to production).
- Deployment Frequency (DF): Reflects the ease or difficulty engineers face when deploying changes for customers.
- Mean Time to Restore (MTTR): Measures the time taken to fix an issue when something breaks, closely related to the next metric.
- Change Failure Rate (CFR): Tracks how often deployments to production result in failure.

Since then, these metrics have gained widespread industry adoption, with annual reports providing updated data through the DevOps Research and Assessment (DORA) [1]. While outcome is the key measure under this umbrella, various technical capabilities are required to achieve desired results. In fact, 24 capabilities have been identified in contexts where outcomes aligned with business expectations, spanning both technical and cultural dimensions. Testing automation is on this list as a critical area of support. A reliable test suite enables confident continuous delivery[19].

More recently, academia has explored the SPACE framework for assessing developer productivity and experience. Findings suggest that productivity encompasses not only technical aspects but also human factors, including[16]: Satisfaction and well-being, Performance, Activity, Communication and collaboration, Efficiency and flow.

Despite insights from both DORA and SPACE, the application of Test-Driven Development (TDD) in professional environments often relies on guidelines and experienced practitioners introducing the practice to software development teams[21]. This can sometimes lead to an uninformed approach to TDD adoption.

4 TDD maturity model

Despite efforts of researchers to create guidelines on TDD, much of the focus has remained on the technical aspects of the practice, while its interaction with other parts of the software development lifecycle has received less attention. TDD is a process that intersects with various areas of software development [22].

TDD impacts the entire software development process in technical aspects as well as in business aspects. It begins with the organization's goals, which are considered when defining a software testing strategy, and extends to the practice

[1] https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

of TDD in a team setting and the generation of test smells. The detailed model with each item for each area is available online[2]. The proposal that follows focuses on areas of software development and processes based on the CMMI framework built taking action research as the basis to create it [6]. The levels of the TDD maturity model are as follows:

- Level 1 - Initial: Interested in the TDD approach, learning about it and curious about how to approach development with a test first, in professional settings, the learning and the delivery happen at the same time.
- Level 2 - Managed: Write tests but not with TDD
- Level 3 - Defined: TDD is used, however, test smells are not tackled
- Level 4 - Quantitatively Managed: TDD is used, and test smells are tackled with a free approach
- Level 5 - Optimizing: Data-driven and continuous refactoring of tests/code - oriented to the value of the test for the business

Building a maturity model for TDD adoption in a business context points towards a conversation between business and technology to incorporate the TDD practice and move into a continuous improvement process that takes into account organizational aspects in addition to technical excellence.

5 The preliminary evaluation

The preliminary evaluation was done in different stages. The first stage was to create materials for practitioners to understand the model and share the background behind it. In addition to that, a Google Group was created to centralize all the materials posted and to have a centralized space for discussion.

5.1 Materials

The materials generated for practitioners to get to know the maturity model were the following: a slide deck with the maturity model content, a thread with the previous work done on this research, the maturity model conception in the format of a paper and lastly, a video with a brief introduction of the research.

5.2 Evaluation

To get a first assessment from real TDD practitioners this study started to look for participants to join the research in popular groups that advocate for TDD and discuss it. Given those requirements, two target groups were contacted:

- Software crafters - Slack has more than 8.000 members[3]

[2] The slideck with detailed information for each area and level is available at https://docs.google.com/presentation/d/1T-1D19gPe5Qp3bhllRfyiyh56wvx3gvk5zhv9r85Uls

[3] https://slack.softwarecrafters.org/

– We do TDD - Slack has more than 800 members [4]

In addition to those groups, the study was shared through X[5][6] and LinkedIn[7]. With that, a total of 14 practitioners joined the Google Group to start the process of evaluation. Once practitioners joined they were presented with a thread with welcome material with the steps of the study and what was expected from them to go through. The second thread created shared the materials mentioned in the section 5.1. This thread specifically received attention from practitioners where they shared feedback. The third thread shared the survey with practitioners as the official channel for offering their perspective of the model. The survey was created and split into four sections:

– Introduction - This section presented materials available to go through before filling out the survey
– About you - This section had general questions about career and TDD practice
– The maturity model proposal - This section had questions focused on the proposal structure and, the levels that the model is split into.
– Closing section - Focuses on the bureaucracy of handling the data provided by them and a confirmation that the data submitted was correct.

5.3 Preliminary results

The entire evaluation was done in three weeks, during the period of evaluation, constant reminders were sent through private chats and the group itself, reminding the practitioners to complete the survey, in total 11 responses were recorded, out of 14 practitioners that joined the evaluation. The raw dataset with sensitive information hidden is available for further scrutiny by the community [8].

About you Practitioners that joined the study ranged from different years of experience, the least amount is 2 years and the maximum 43. Most of them are working on backend applications, however, some of them are focused on the frontend in addition to the backend. The age of practitioners corroborated with the years of experience. More than 50% of practitioners are between 31 and 50 years old.

The industry varies with most of them with experience with more than one industry. Healthcare, startups and banking are the top 3 industries, most of

[4] https://wedotdd.com
[5] The first published message sharing the research on X https://x.com/MatheusMarabesi/status/1811126885839196660
[6] Second attempt https://x.com/MatheusMarabesi/status/1812131152724914274
[7] Published post on LinkedIn inviting practitioners https://www.linkedin.com/posts/marabesi_tdd-activity-7216896466986930176-bMD3
[8] The dataset with responses of practitioners are available at https://gist.github.com/marabesi/283775271fac624398291bc61829ad97

practitioners reported working experience in those. Other industries such as hotel, video/streaming and telecom appeared once. Regarding company sizes, more than 60% practitioners have worked for large companies.

Focusing on the TDD practice in the industry, the practitioners of the study showed that they have an amount of experience that is less in comparison with their years of experience, which suggests that they started working in the software industry before using TDD making the transition to a test first afterward. The maximum years of experience doing TDD for professional projects was 24 years, whereas the maximum years of experience in the industry was 43. Java, C#, Kotlin, Typescript and Go are the top 5 programming languages used by practitioners. However, responses covered a wide range of languages, Python, Clojure and C++ to name a few.

The maturity model proposal In this first contact with the model by experienced practitioners the survey dedicated a section focusing on the applicability and the problem space of the model. More than 70% of practitioners agreed that adopting TDD in a professional setting is challenging. On the same trend, more than 60% practitioners agreed that the maturity model proposed might provide useful guidance.

The basis for this maturity model is the intersection with other disciplines of TDD such as domain knowledge, agile culture, requirements and software design, it was agreed by all practitioners that those areas intersect with TDD. In addition to that, 5 levels are an appropriate number of levels for more than 70% of practitioners. The same positive trend was found in the areas of the model. More than 60% agreed that the areas Requirements, Technical enabler and Teams settings are appropriate for the proposed model, the same was found for the distribution of items between those areas.

Practitioners agreed that the proposed model offers an incremental approach for TDD adoption in a software development team inside an organization. The incremental approach is one of the aspects offered by the CMMI as well. Practitioners agreed that the context of the organization influences the TDD adoption.

Discussion For the maturity model itself, results point to a model that is useful for adopting TDD in professional settings, however, results for each level point that at its current form improvements are needed. The model might have a well-rounded proposal about areas and subjects that intersect with TDD. Therefore, despite not being detailed here, practitioners offered feedback regarding the implementation of the levels reporting that they need adjustments before use in real projects[9].

For level 1, tasks that have technical details might not affect the TDD flow. For example, in level 2 the discrepancy of the data points that the deployment date might not be that important for TDD practitioners, however it is for businesses, regarding code coverage, practitioners disagreed that code coverage is the only

[9] The feedback is available in the dataset as previously mentioned in this study.

way of measuring the test code, further investigation is required to explore alternatives to incorporate in the model.

In the next section, we present the follow up study that was done with experts in the field of TDD to evaluate the model and its levels.

6 Evaluation by experts

This section outlines the process of validating the TDD maturity model framework using expert judgment through a systematic feedback from practitioners.

6.1 Methodology

The primary objective of the research was to validate a comprehensive instrument designed to evaluate the adoption of Test-Driven Development (TDD) in professional settings. The validation process focuses on examining the degree to which the instrument encompasses the relevant and representative elements of the TDD practice [2]. To that end, the work by [18] [13] [25] was used as a base for the methodology presented in this section using the content validity and expert judgment.

6.2 Participants

The participants in this study are practitioners with experience in TDD applying the practices in their daily work in different contexts. In total 10 practitioners recorded their evaluation (out of 16 participants invited, 10 completed the evaluation.), they were selected based on their experience and knowledge of TDD. Most of participants have more than five years of experience in the industry and have worked in different companies and projects applying TDD. The participants were invited through, coderetreat Madrid (event in person), invited directly through social media, invited by email (suggested by others participants in the study) and invited from the Okiwi craft community.

6.3 Data Collection

The data collection was done exclusively via Google Docs, where the participants were asked to evaluate the TDD maturity model[10][11]. The participants were constantly reminded through email to complete the evaluation.

[10] The template used to collect the data is available at https://docs.google.com/spreadsheets/d/1hyO-hSpnNzd8-_Zmad2J7tV-T6Acb0sWf8Uapi8ZKhw

[11] The detailed analysis is available for the community at https://marabesi.github.io/tdd-maturity-model-artifacts

6.4 Evaluation results

For each level shared with participants, it was asked about a general overview of the model in four criteria, named: Clarity, Organization, Sufficiency and Coherence. While the mean and median values indicate a generally positive perception of the levels, the variability measures (SD and CV) reveal areas where opinions are more divided, highlighting potential opportunities for refinement in specific levels.

The mean ratings for clarity (Table 1) across all levels range from 3.5 to 3.9, with the median consistently at 4, indicating that participants generally found the descriptions understandable. However, the highest SD (0.972) and CV (27.77%) occur at Level 4, suggesting that clarity at this stage is inconsistently perceived.

Level	Std Dev	CV	Mean	Median	Min	Max
1	0.699	19.42 %	3.6	4	2	4
2	0.316	8.11 %	3.9	4	3	4
3	0.316	8.11 %	3.9	4	3	4
4	0.972	27.77 %	3.5	4	1	4
5	0.483	13.06 %	3.7	4	3	4

Table 1. Clarity (if the level is formulated using appropriate language)

Based on the data provided by participants, two of them reported difficulties in the measurements and practices that the level proposes. Leading to a decreased clarity on the the items provided. In contrast, Levels 2 and 3 show the lowest SD (0.316) and CV (8.11%), indicating a strong agreement among participants.

The organization (Table 2) criterion maintains relatively stable mean scores between 3.6 and 3.9, with median values at 4. The highest variability is at Level 2 (SD = 0.699, CV = 19.42%).

Level	Std Dev	CV	Mean	Median	Min	Max
1	0.516	14.34 %	3.6	4	3	4
2	0.699	19.42 %	3.6	4	2	4
3	0.316	8.11 %	3.9	4	3	4
4	0.632	16.64 %	3.8	4	2	4
5	0.422	11.10 %	3.8	4	3	4

Table 2. Organization (there is a logical organization between the elements of the level)

From the data gathered from participants point that the organization of the model should support the learning of TDD on earlier stages, as it starts with that in level, taking into account how it would impact later levels. In contrast, Level 3 has the lowest SD (0.316) and CV (8.11%), indicating strong agreement.

Sufficiency (Table 3) scores remain fairly stable across levels, with means ranging from 3.5 to 3.8 and medians at 4, suggesting that participants largely agree that the levels contain necessary elements.

Level	Std Dev	CV	Mean	Median	Min	Max
1	0.527	15.06 %	3.5	3.5	3	4
2	0.422	11.10 %	3.8	4	3	4
3	0.483	13.06 %	3.7	4	3	4
4	0.675	18.24 %	3.7	4	2	4
5	0.707	20.20 %	3.5	4	2	4

Table 3. Sufficiency (the level includes the necessary elements in terms of quantity and quality)

However, Level 5 exhibits higher variability (SD = 0.707, CV = 20.20%). From the data, a participant shared that from his perspective there is an expectation that the model would drive towards other aspects of software development, such as architecture.

The coherence (Table 4) criterion sees the most variation among responses, with the lowest mean rating (3.2 at Level 5) and the highest CV (28.72%) across all categories. The minimum score at Level 5 is 1, indicating that some participants found it completely incoherent or misaligned with expectations.

Level	Std Dev	CV	Mean	Median	Min	Max
1	0.516	15.19 %	3.4	3	3	4
2	0.699	20.56 %	3.4	3.5	2	4
3	0.707	20.20 %	3.5	4	2	4
4	0.675	18.24 %	3.7	4	2	4
5	0.919	28.72 %	3.2	3	1	4

Table 4. Coherence (there is a relationship between the different elements that make up the level)

Level 4 also shows moderate disagreement (SD = 0.675, CV = 18.24%), though not as extreme as Level 5. From the data gathered a participant shared that despite the uderstanding of the elements and how they are grouped, there is a lack of showing how the elements communicate between each other and how the progress among levels should work.

6.5 Discussion

The mean ratings for clarity ranged from 3.5 to 3.9, with a median consistently at 4, indicating that participants generally found the descriptions understandable.

However, Level 4 had the highest variability (SD = 0.972, CV = 27.77%), suggesting that clarity at this level was more uncertain.

The mean ratings for organization were relatively stable across levels, with means ranging from 3.5 to 3.8 and medians at 4, indicating that participants largely agreed on the organization of each level. Level 3 had the lowest variability (SD = 0.483, CV = 13.06%), suggesting strong agreement.

The mean ratings for sufficiency were generally stable across levels, with means ranging from 3.5 to 3.8 and medians at 4, indicating that participants largely agreed on the sufficiency of each level. Level 5 had higher variability (SD = 0.707, CV = 20.20%), suggesting disagreement.

The mean ratings for coherence varied significantly across levels, with a low mean rating at Level 5 (3.2) and high variability (SD = 0.919, CV = 28.72%). This suggests that participants had differing opinions on the coherence of each level, particularly at Level 5.

7 Conclusion

In this study, we presented two studies done with practitioners to validate a maturity model for Test-Driven Development (TDD) adoption in professional settings. The first study, was a preliminary evaluation with 11 practitioners, who provided feedback on the model's structure and in each level using a qualitative approach with practitioners. The second, based on the existing literature for expert jugdment went one step further to evaluate each area in four criteria (clarity, organization, sufficiency and coherence) and each level in two criteria (understandable and relevant) through a quantitative approach.

The results from them highlights both the potential and limitations of our maturity model for TDD adoption. While participants generally found each level understandable and organized (as reflected in the stable mean ratings for clarity, organization, and sufficiency), there were significant variations in ratings at certain levels, particularly Level 4 and Level 5.

At Level 4, variability in ratings for clarity was higher than expected, suggesting that participants may have had differing opinions on what constitutes effective TDD practices in a business context. This underscores the need to further investigate and refine our model's proposals about how businesses can leverage TDD to drive code coverage and other key metrics.

At Level 5, the low mean rating for coherence and high variability suggest that participants had differing opinions on how to effectively measure and evaluate TDD adoption at this advanced level. This finding highlights the importance of ongoing research to develop a more comprehensive and coherent understanding of what it means to achieve TDD maturity in professional settings.

More broadly, our results underscore the need for continued refinement and iteration on our maturity model. By acknowledging and addressing these limitations, we can create a tool that is more effective at supporting practitioners as they adopt and improve their TDD practices.

References

1. Abushama, H.M., Alassam, H.A., Elhaj, F.A.: The effect of test-driven development and behavior-driven development on project success factors: A systematic literature review based study. In: 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). pp. 1–9 (2021). https://doi.org/10.1109/ICCCEEE49695.2021.9429593
2. Almanasreh, E., Moles, R., Chen, T.F.: Evaluation of methods used for estimating content validity. Research in social and administrative pharmacy **15**(2), 214–221 (2019)
3. Aniche, M.: Effective Software Testing: A Developer's Guide. Simon and Schuster (2022)
4. Aniche, M.F., Gerosa, M.A.: Most common mistakes in test-driven development practice: Results from an online survey with developers. In: 2010 Third International Conference on Software Testing, Verification, and Validation Workshops. pp. 469–478. IEEE (2010)
5. Anwer, F., Aftab, S., Waheed, U., Muhammad, S.S.: Agile software development models tdd, fdd, dsdm, and crystal methods: A survey. International journal of multidisciplinary sciences and engineering **8**(2), 1–10 (2017)
6. Avison, D.E., Lau, F., Myers, M.D., Nielsen, P.A.: Action research. Communications of the ACM **42**(1), 94–97 (1999)
7. Beck: Test driven development: By example (2002)
8. Beck, K., Gamma, E.: Test infected: Programmers love writing tests. Java Report **3**(7), 37–50 (1998)
9. Benato, G.B., Vilela, P.R.S.: Test-driven development: uma revisão sistemática. Revista Brasileira de Computação Aplicada **13**(1), 75–87 (mar 2021). https://doi.org/10.5335/rbca.v13i1.11154, https://seer.upf.br/index.php/rbca/article/view/11154
10. Bissi, W., Serra Seca Neto, A.G., Emer, M.C.F.P.: The effects of test driven development on internal quality, external quality and productivity: A systematic review. Information and Software Technology **74**, 45–54 (2016). https://doi.org/https://doi.org/10.1016/j.infsof.2016.02.004, https://www.sciencedirect.com/science/article/pii/S0950584916300222
11. Causevic, A., Sundmark, D., Punnekkat, S.: Factors limiting industrial adoption of test driven development: A systematic review. In: 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. pp. 337–346. IEEE (2011)
12. Chaudhary, M., Chopra, A.: CMMI for development: Implementation guide. Apress (2016)
13. Escobar-Pérez, J., Cuervo-Martínez, Á.: Validez de contenido y juicio de expertos: una aproximación a su utilización. Avances en medición **6**(1), 27–36 (2008)
14. Fielding, R.T., Taylor, R.N., Erenkrantz, J.R., Gorlick, M.M., Whitehead, J., Khare, R., Oreizy, P.: Reflections on the rest architectural style and "principled design of the modern web architecture" (impact paper award). In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. p. 4–14. ESEC/FSE 2017, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3106237.3121282, https://doi.org/10.1145/3106237.3121282
15. Forsgren, N., Humble, J., Kim, G.: Accelerate—building and scaling high performing technology organizations (The science of lean software and DevOps). IT Revolution Press Portland (2018)

16. Forsgren, N., Storey, M.A., Maddila, C., Zimmermann, T., Houck, B., Butler, J.: The space of developer productivity: There's more to it than you think. Queue **19**(1), 20–48 (mar 2021). https://doi.org/10.1145/3454122.3454124, https://doi.org/10.1145/3454122.3454124
17. Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., Juristo, N.: A dissection of the test-driven development process: Does it really matter to test-first or to test-last? IEEE Transactions on Software Engineering **43**(7), 597–614 (2016)
18. Gutiérrez-Pérez Bárbara Mariana, M.G.A.V.: Content validation of an instrument used to assess the educational quality of blended learning courses. Journal of Hunan University Natural Sciences **48**(10) (2021)
19. Humble, J., Farley, D.: Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education (2010)
20. Kurkela, M.: Devops capability assessment in a software development team (2020)
21. Law, W.K.A.: Learning effective test driven development https://www.scitepress.org/PublishedPapers/2006/13161/13161.pdf
22. Marabesi, M., García-Holgado, A., García-Peñalvo, F.J.: Exploring the connection between the tdd practice and test smells—a systematic literature review. Computers **13**(3) (2024). https://doi.org/10.3390/computers13030079, https://www.mdpi.com/2073-431X/13/3/79
23. Meszaros, G.: xUnit test patterns: Refactoring test code. Pearson Education (2007)
24. Munir, H., Moayyed, M., Petersen, K.: Considering rigor and relevance when evaluating test driven development: A systematic review. Information and Software Technology **56**(4), 375–394 (2014)
25. Revista de Pedagogía, B.: Vol. 73, nº 2, 2021. Bordón. Revista de Pedagogía **73**(2) (jul 2021), https://recyt.fecyt.es/index.php/BORDON/article/view/90572
26. Pedroso, B., Jacobi, R., Pimenta, M.: Tdd effects: Are we measuring the right things? In: Agile Processes in Software Engineering and Extreme Programming: 11th International Conference, XP 2010, Trondheim, Norway, June 1-4, 2010. Proceedings 11. pp. 393–394. Springer (2010)
27. Rafique, Y., Mišić, V.B.: The effects of test-driven development on external quality and productivity: A meta-analysis. IEEE Transactions on Software Engineering **39**(6), 835–856 (2013). https://doi.org/10.1109/TSE.2012.28
28. Richardson, L.: Justice will take us millions of intricate moves. In: QCon conference (2008), https://www.crummy.com/writing/speaking/2008-QCon
29. Staegemann, D., Volk, M., Pohl, M., Haertel, C., Hintsch, J., Turowski, K.: Identifying guidelines for test-driven development in software engineering—a literature review. In: Yang, X.S., Sherratt, S., Dey, N., Joshi, A. (eds.) Proceedings of Seventh International Congress on Information and Communication Technology. pp. 327–336. Springer Nature Singapore, Singapore (2023)
30. Tarlinder, A.: Developer testing: Building quality into software. Addison-Wesley Professional (2016)
31. Yahya, N., Awang Abu Bakar, N.S.: Test driven development contribution in universities in producing quality software: A systematic review. In: The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M). pp. 1–6 (2014). https://doi.org/10.1109/ICT4M.2014.7020666
32. Zarour, M., Alhammad, N., Alenezi, M., Alsarayrah, K.: A research on devops maturity models. Int. J. Recent Technol. Eng **8**(3), 4854–4862 (2019)