Sentiment Overflow in the Testing Stack: Let the Finger Pointing Begin

Mark Swillus
Software Engineering Research Group
Delft University of Technology
Delft, The Netherlands
m.swillus@tudeflt.nl

***Abstract*—We know that many software projects do not adopt testing practices [2]. But we still lack understanding of why practitioners do not test. I argue that filling this knowledge gap is required in order to take the right steps to tackle the lack of testing. I even go further and challenge tool developers and advocates of testing practices to reflect on how their effort is addressing this "why?". If efforts are made without such reflection, they can do more harm than good. The preliminary findings of my research show that we can gain this understanding by investigating the social world in which testing is practiced.**

***Index Terms*—software testing, reddit, stack overflow, qualitative research**

I. Investigating social aspects of testing

Researchers in the field of software engineering can greatly learn from studies in other fields that apply theories and methods from social psychology. Multiple studies have shown for example, that policies and the creation of incentives to promote good behavior can have counterintuitive effects. Introducing penalties or sanctions for *bad* behavior can for example lead to even worse behavior [5]. Providing feedback to encourage individuals to behave *good*, can cause the opposite as well [11; 3]. Incentivizing good behavior is difficult and there is usually no *silver-bullet* [4]. Goal framing theory states that every choice we make is evaluated using different goal frames [8; 6]. When waiting at a red traffic light, the presence of a child causes my *normative* or moral goal frame (I want to lead by example), to be dominant over my *gain* goal frame (I could save 2 minutes of my time), so I decide to wait for the light to turn green. Similarly, a decision not to write a unit test is unlikely to be based on one single factor. The answer to the question of whether or not the investment of time to write a test is worth it in comparison to testing manually, is never *the only* factor that leads to the decision (not) to test, nor is it the excitement one feels when using a nifty tool. Crucially, using a *silver-bullet* solution (like TDD in the eyes of some) in the wrong situation can lead to the opposite of what was intended. What a researcher, developer or manager with limited understanding of the social world of practitioners considers to be an incentive for testing could in practice lead to discouragement.

This research was partially funded by the Dutch science foundation NWO through the Vici "TestShift" grant (No. VI.C.182.032).

I hypothesize that the decision not to test software is highly dependent on the social environment that software engineering is practiced in. Documenting and understanding these environments is required to develop truly effective recommendations and incentives for testing. Two studies which I have conducted in the past year take a step into this direction by discerning the perspectives of testers. I qualitatively analyzed posts on Stack Overflow and reddit to learn what affects practitioners when they practice testing. The first study on Stack Overflow revealed that when practitioners express themselves in a sentimental way, they are either discouraged from practice or aspire for more than getting the job done. Negative sentiment is expressed by practitioners who are ambiguous about practice and when tools, which are used in very unique development environments, cause unexpected behavior. On the other side of the spectrum, confidence and the understanding of long term goals is what practitioners report in posts with positive sentiment. Further, the analysis of Stack Overflow suggests, that it is common to only start testing when the complexity of projects renders manual testing impossible. Instead of learning testing practices gradually, practitioners throw themselves into the breach when it is too late for simple, approachable solutions. On reddit one can read, among other things, reports of toxic behavior of developers which is surfacing in the context of testing. As testing can reveal mistakes of others, it has the potential to cause conflict. Instead of cooperating, some *start pointing fingers* or *kill the messenger*. What reddit and Stack Overflow both reveal is the importance of normative goals to developers when deciding to use testing practices. They want to what is appropriate. This has also been implied by others in different contexts [7; 1; 10; 9]. But the analysis of reddit and Stack Overflow also shows that even if developers are sensitive to appropriateness, it is likely that more selfish motives take over when they do not know how to act appropriately or when they do not receive the right support. I conclude that by discerning and understanding perspectives of developers, including their selfish and normative goals, we will be able to create truly meaningful and valuable recommendations and tools. I thus want to motivate others to explore and share insights about the social world of developers in which those perspectives take shape. Let us be attentive when there is an overflow of sentiment and let us observe closely when fingers are pointed.

References

[1] Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. 2008. Motivation in Software Engineering: A systematic literature review. *Information and Software Technology* 50, 9-10 (Aug. 2008), 860–878. https://doi.org/10.1016/j.infsof.2007.09.004

[2] Moritz Beller, Georgios Gousios, Annibale Panichella, Sebastian Proksch, Sven Amann, and Andy Zaidman. 2019. Developer Testing in the IDE: Patterns, Beliefs, and Behavior. *IEEE Transactions on Software Engineering* 45, 3 (March 2019), 261–284. https://doi.org/10.1109/TSE.2017.2776152

[3] Gwendolyn Brandon and Alan Lewis. 1999. Reducing Household Energy Consumption: a Qualitative and Quantitative Field Study. *Journal of Environmental Psychology* 19, 1 (March 1999), 75–85. https://doi.org/10.1006/jevp.1998.0105

[4] Ernst Fehr and Armin Falk. 2002. Psychological foundations of incentives. *European Economic Review* 46, 4-5 (May 2002), 687–724. https://doi.org/10.1016/S0014-2921(01)00208-2

[5] Uri Gneezy and Aldo Rustichini. 2000. A Fine is a Price. *The Journal of Legal Studies* 29, 1 (Jan. 2000), 1–17. https://doi.org/10.1086/468061

[6] Peter Gollwitzer and Gordon Moskowitz. 1996. Goal Effects on Action and Cognition. *First publ. in: Social psychology: Handbook of basic principles. New York: Guilford, 1996, pp. 361-399* (Jan. 1996).

[7] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un)happy. *Journal of Systems and Software* 140 (June 2018), 32–47. https://doi.org/10.1016/j.jss.2018.02.041

[8] Siegwart Lindenberg and Linda Steg. 2007. Normative, Gain and Hedonic Goal Frames Guiding Environmental Behavior. *Journal of Social Issues* 63, 1 (March 2007), 117–137. https://doi.org/10.1111/j.1540-4560.2007.00499.x

[9] André Meyer, Earl T Barr, Christian Bird, and Thomas Zimmermann. 2021. Today was a Good Day: The Daily Life of Software Developers. (2021). https://doi.org/10.5167/UZH-170375

[10] Helen Sharp, Nathan Baddoo, Sarah Beecham, Tracy Hall, and Hugh Robinson. 2009. Models of motivation in software engineering. *Information and Software Technology* 51, 1 (Jan. 2009), 219–233. https://doi.org/10.1016/j.infsof.2008.05.009

[11] Jeannet H. van Houwelingen and W. Fred van Raaij. 1989. The Effect of Goal-Setting and Daily Electronic Feedback on In-Home Energy Use. *Journal of Consumer Research* 16, 1 (June 1989), 98. https://doi.org/10.1086/209197