On personality testing and software engineering

Clayton Lewis
Coleman Institute for Cognitive Disabilities
University of Colorado
clayton.lewis@colorado.edu

There is current interest in applying methods from the psychology of personality to software engineering. For example, in a recent review, Graziotin et al. (2020) say,

> "Software engineers are knowledge workers and have knowledge as main capital [98]. They need to construct, retrieve, model, aggregate, and present knowledge in all their analytic and creative daily activities [89]. Operations related to knowledge are cognitive in nature, and cognition is influenced by characteristics of human behavior, including ***personality, affect, and motivation*** [47]. It is no wonder that industry and academia have explored psychological aspects of software development and the assessment of ***psychological constructs at the individual, team, and organization level*** [30, 61]. [emphasis added.]

They are quite critical of much current work, including that using the discredited Myers-Briggs Type Indicator (MBTI, which remains popular in the literature), and other work that attempts to adapt what the authors feel are well validated methods from other areas of social science in ways that are scientifically unsound. The authors then present what they see as the methods needed for sound work in the area, emphasizing that doing the work correctly requires an enormous amount of work. They conclude,

> "The software engineering community must value psychometric studies more. This, however, requires a cultural change that we hope to champion with this paper. 'Spending an entire Ph.D. candidacy on the validation of one single measurement of a construct should be, not only approved, but encouraged.' [43] and, we believe, should also become normal."

We are skeptical that the psychometric methods the authors advocate can usefully be applied in software engineering. Further, we see the potential for substantial harm from efforts to apply them.

As an illustration of the issues, we'll use the hypothetical psychometric study included as a methodological appendix to Graziotin, et al. (2020). The study aims to illuminate "individual perception styles of source code":

> "Our fictitious construct is the 'individual perception styles of source code' that, through a literature review (or, perhaps, after a grounded theory study), we believe is mainly composed of, or highly related, to the following five constructs: code curiosity, programming paradigm flexibility, learning disposition, collaboration propensity, and comfort in novelty."

Some premises of such an investigation are that there is a population of source code perceivers whose attributes can be studied, and that these individuals have characteristics like "code curiosity" or "learning disposition" that can usefully be measured. The nature of psychometric methods requires that investigators have access to many members of the population to be studied.

The nature of software engineering is that it is complex in a great many different ways, that all influence "source code perception". What tools are being used to examine source code? What language is the code in (how does perception of Python code relate to perception of an Excel macro)? Is the code being viewed in the context of coding, or debugging, or testing, or maintenance? Is the perceiver already familiar with the code base, or are they a new team member? Are they experienced or inexperienced in the particular task being examined? Are they working individually, or in a group code review? We think it highly unlikely that enough data could be gathered in any one of the situations defined by the cross product of these distinctions to support a proper psychometric analysis. That is to say, undertaking a psychometric analysis of this construct really means that one thinks that "source code perception" isn't actually influenced by these things in important ways.

Does this argument mean that all empirical study of phenomena as complex as software engineering is futile? We think it does mean that empirical study that is *purely* empirical cannot succeed. That is, research has to be guided by ideas about the mechanisms at work in the phenomenon, mechanisms that can be identified in action across varying situations. Cognitive dimensions analysis is an example of a framework that can offer that (see Lewis, 2017, for related discussion).

We have concerns about the constructs proposed as "related" to source code perception, as well. To undertake this psychometric investigation means that one believes that things like "programming paradigm flexibility" are stable characteristics of people. But given the complexity of software engineering situations, and of people, we see no reason to expect that. Rather, we expect that people's behavior, and preferences, will be influenced by the tools they use, their past experience, their current social environment, their personal goals, and much more. There just isn't such a thing as "programming paradigm flexibility" independent of these contextual factors.

We understand that the constructs in this example from Graziotin et al. are made up, so it is pointless to argue about the specifics of them. But we agree with Graziotin et al. that they illustrate the logic of many studies that might actually be undertaken, or have been undertaken.

An advocate of psychometric research might respond, "Well, if you are right, our studies will fail, but if you are wrong, our studies will reveal that there really are these stable characteristics of software engineers, with consequences that cut across the distinctions among situations that you are concerned about. So you should let us get on with the work, and we'll see who is right."

Of course, people are free to investigate whatever they like, and we can't forbid them, even if we wanted to. But we can register our doubts that the large investments needed will prove justified. In particular, we strongly dissent from the suggestion that "Spending an entire Ph.D. candidacy on the validation of one single measurement of a construct should be, not only approved, but encouraged." We would certainly not encourage our students to do such a thing.

Our concerns go farther. The persistence of the discredited MBTI in the literature suggests that psychometric ideas are *seductive*. People often want to believe that there are simple, knowable characteristics of people that will allow us to deal more straightforwardly and objectively with our fellow creatures. Rather than embracing human diversity, we often want to reduce people to numbers: we should hire this candidate rather than this one, because they scored better on this assessment, or we should assign this person this role rather than that one, because they show higher "programming paradigm flexibility".

That seems easier than trying to assess their job experience. The appeal of this vision is such that it often overrides methodological scruples, as Graziotin et al. document. The research community needs to push back.

We think that some potential applications of psychometrics to software engineering are actually *harmful*, as well as wrong, adding urgency to the push back. The belief that software engineers have stable psychometric characteristics that we can measure, and that predict their aptitude for particular roles, leads naturally to making decisions about hiring, or promotions, on the basis of these measurements. Indeed, Graziotin et al. envision just these applications, and the potential for harm accompanying them:

> "Improper development, administration, and handling of psychological tests could harm the company by hiring a non-desirable person, and it could harm the interviewee because of missed opportunities."

For Graziotin et al., these potential harms call for ***proper*** "development, administration, and handling of psychological tests." But we argue that ***there can be no such thing*** as proper development and application of such tests, because of the complexity of people, software engineering situations, and their combinations.

Acknowledgement. I thank Luke Church for useful comments.

References

Graziotin, D., Lenberg, P., Feldt, R., & Wagner, S. (2020). Behavioral Software Engineering: Methodological Introduction to Psychometrics. arXiv preprint arXiv:2005.09959.

Lewis, C. (2017). Methods in user oriented design of programming languages. In Proc. PPIG 2017- 28th Annual Workshop. Online at https://ppig.org/papers/ .