Process Theories and Taxonomies in Software Engineering

Aftab Hussain
University of California, Irvine
aftabh@uci.edu
February 2, 2020

This article presents notes taken from Paul Ralph's 2018 paper [1]. The paper proposes guidelines synthesized on the basis of process theories and taxonomies that can enable practitioners to understand "how" things happen in the software engineering domain. Most existing theories, as mentioned by the author, illustrate the "why" aspect in this regard. Here, key ideas of the paper are highlighted, using a casual writing style.

Guidelines have a broad scope?

The introduction of the paper discusses a dichotomy of software engineering research thought. The first are theories, which are used "for explaining, predicting, analyzing and understanding diverse SE phenomena". Three kinds of theories are clearly and concisely described: variance theories, process theories, and theories for understanding. (As a sidenote, from these definitions it can be inferred that most of the hypotheses explored in the software repository mining community, e.g. [2,3], fall under the umbrella of variance theories.) The second component of the dichotomy are taxonomies.

The focus of this paper is on process theories and taxonomies. The purpose of the paper is stated as, "to adapt existing guidance on developing and evaluating taxonomies and process theories for software engineering".

An equivalency between process theories and taxonomies is argued, and consequently the author highlights the need to study them together. Before embarking on discussing the equivalency of the two, the author discusses the naivety in characterizing process theories as not real theories and the subtleties of classification (a clearly explained student grouping example is used.).

Details of what is the scope of these guidelines, and what is meant by the term "guidance (or guideline)" itself are not elaborated upon in the introduction. However, we get some insight from this point proposed as one of the reasons for coming up with such a guideline: "Reading a single introduction, tailored to one's field, is more efficient and less confusing than wading through a hundred papers and books from a dozen other fields trying to create and justify a reasonable methodology." A very salient reality often encountered in trying to understand a research area.

Yes, the scope of the guidelines is broad

In the section where the importance of process theories and taxonomies are discussed (Why are taxonomies and process theories important?), it becomes clear that the author seeks to attack a very broad problem in all software engineering. Some excellent points are shown on the narrowness of existing guides for (such as those derived from the field of software development methodologies) explaining software engineering discipline. The author's reasoning substantiates a strong position for the need of having more fine-grained terminology for describing various aspects of software engineering, which the author believes could be obtained from process theories and taxonomies.

Good resource on process theories

The section, "A Brief Review of Process Theories" gives a good background on work in process theories in the area of management. It expands on Van de Ven and Poole's classifications of types of process theories [4] — *dialectic*, *teleological*, *evolutionary*, and *lifecycle*. It also elaborates on some misconceptions on process theories, such as,

1. Process theories are not software development methodologies like Scrum and Lean [5].

2. A process theory is not an algorithm.
3. Causal relationships between variables cannot be typically drawn from process theories.

Finally, the section also provides an elaboration on the quality aspect of process theories, and their limitations.

13 Interesting Takeaways

(Please refer to the paper for citations in the following quotes.)

1. Rethinking how we think about qualitative research:
"rejecting qualitative process theory research for having small, convenience samples while accepting experiments with small, convenience samples is prejudice against qualitative research, not "high standards."

2. Process theory has measurability issues:
"They normally do not make point estimate predictions; for example, how long a process will take. With the exception of evolutionary theories, process theories rarely make probabilistic predictions; for example, whether a project will succeed."

3. Improving the state-of-the art based on process theories is difficult, since process theories don't generally recommend:
"incorporating prescriptions into a process theory is intrinsically problematic. While a researcher might make prescriptions by interpreting a situation using a process theory, the theory itself should explain only what is, not what should be."

4. Following from the above, building theories from software development methodologies is tricky because of their inherent difference in utility:
"An SDM (software development method) is a system of prescriptions for doing something effectively. Methods are inappropriate foundations for taxonomies and process theories … Methods prescribe what should exist and should be done. Taxonomies and process theories describe what does exist and what actually happens. Adapting a method into a theory therefore tends to over-rationalize reality …"

5. The importance of being cautious of theorizing based on personal experience:
"theorizing by reflecting on your own experience is hazardous because our experiences are misleading. They are rarely a representative sample. Our beliefs are often not calibrated to available evidence …our memories often diverge from actual events…."

6. Expertise cannot be relied upon solely:
"Our expertise does not inoculate us against cognitive and perceptual biases … experts are susceptible to system justification—the tendency to irrationally defend the status quo …Researchers are also prone to over-rationalizing phenomena …The entire point of Grounded Theory is, arguably to overcome these problems by holding the researcher close to the data … "

7. Some useful questions to address when developing a process theory:
"What entities are changing?"
"Is there a pattern at all?"
"If there is a pattern, what type is it?"

8. Tips for writing a good software engineering theory paper:
"the paper should describe the emergence of the theory. While the theory generation process may involve non-replicable intuitively leaps, the reader should be able to recover and understand the researcher's logic and process. In my experience, the literature review or empirical study, not the theory itself, often carries the paper and convinces the reviewers"

9. *"Guidelines are supposed to help, not shackle."*

10. *"There is no such thing as representative sampling in case studies or grounded theory."*

11. Theory paper reviewers should *"not expect a manuscript to review all relevant prior work, because that could include thousands of studies"*

12. Theory paper reviewers should "*expect explicit definitions for all elements of the proposed theory*"

13. Theory paper reviewers should "*not criticize a theory for lacking novelty unless you can cite an existing theory that explains the same phenomenon better than the proposed theory. Researchers may avoid emphasizing novelty because peer review is biased against new ideas*"

References

[1] Paul Ralph, Toward Methodological Guidelines for Process Theories and Taxonomies in Software Engineering, IEEE Transactions on Software Engineering, 2018

[2] Di Yang, Aftab Hussain, Cristina Videira Lopes, From query to usable code: an analysis of stack overflow code snippets, Proceedings of the 13th International Conference on Mining Software Repositories, 2016

[3] Stack Overflow in Github: Any Snippets There?, Proceedings of the 14th International Conference on Mining Software Repositories (MSR), 2017

[4] A. H. Van de Ven and M. S. Poole, "Explaining development and change in organizations," The Academy of Management Review, vol. 20, no. 3, pp. 510–540, 1995.

[5] M. Poppendieck and T. Poppendieck, Lean Software Development: An Agile Toolkit. Addison-Wesley Professional,
2003.