Soft Theory: A Pragmatic Alternative to Conduct Quantitative Empirical Studies

Daniel Russo and Klaas-Jan Stol
Lero—the Irish Software Research Centre
School of Computer Science and Information Technology
University College Cork
daniel.russo@lero.ie, klaas-jan.stol@lero.ie

***Abstract*—Practitioners and scholars often face new software engineering phenomena which lack sufficient theoretical grounding. When studying such nascent and emerging topics, it is important to establish an initial and rudimentary understanding, leaving a more precise understanding of underpinning mechanisms till later. Controlled experiments, for example, might lead to insights into the specific mechanisms underpinning a certain practice, such as distributed development, pair programming, and test-driven development. However, at an initial stage of research, such highly controlled studies may not be feasible. In other domains, it may not be clear what the key constructs are, so that effective measurement cannot be done. Instead, researchers might opt for pragmatic alternative research approaches that do not require experimental control or active intervention in a study's setting. In this paper we advocate the use of soft theory (based on soft modeling techniques) for quantitative studies in software engineering research. We discuss the use of soft theory and position it within an existing taxonomy of quantitative data analysis techniques. Soft modeling and soft theory affords us a pragmatic approach to developing inferential and predictive research models, rather than aiming to develop a causal understanding. Soft theory approaches are grounded in robust quantitative data analysis techniques. We argue that these techniques can be effectively used in industry settings which are not amenable to highly controlled studies.**

I. Introduction

Traditionally, the development and use of theory has not been a primary activity within Software Engineering (SE) research [1], [2]. The lack of a rigorous theory or conceptual model is the default scenario in engineering disciplines, where professional practice tends to advance before a theoretical foundation is developed. An oft-cited reason is that SE is still a young discipline whose roots can be traced back to the 1950s and 1960s. As Margaret Hamilton explained in her ICSE 2018 keynote, marking the 50th anniversary of Software Engineering, most knowledge and wisdom that we possess come from trial and error experiences in practice [3]. Of course, this is not to trivialize decades of research that the community has conducted, some of which has had a significant impact on the field and the community. For example, *Conway's Law* [4] suggests that software design is affected by an organization's communication structure; or *Brooks' Law* [5] which has implications for project management of software projects that are running late; or the 201 principles of programming by Davis [6]. A common theme in these examples is that the sometimes anecdotal evidence that is offered is generalized, without sufficient theoretical and empirical justification.

An example of practice leading research is design patterns. Many SE students are taught about design patterns as reusable design solutions to recurring problems, and that they can draw design ideas from catalogues of well-defined problem-solution pairs. This concept was adopted by software professionals from the civil architecture domain [7], not proposed by software engineering researchers. Although the typical software engineer may rely on some SE theory[1] every day, such as design patterns, theory is still frequently overlooked in software engineering research. For example, a survey study of 103 articles reporting experiments selected from of a total of 5,453 articles published in major SE research venues, shows that only 24 used some theoretical basis to inform the research [8].

While there have been some efforts within the SE domain to establish a more clear focus on theory [2], the SE community has not widely adopted a theory-oriented approach to research. Especially in nascent areas of research, theory underpinning phenomena and events is usually lacking. We may have a *conceptual* idea of the key theoretical constructs, but it would be too early to aim for development of a complete theory.

In this paper, we draw a distinction between "hard" and "soft" theory. We characterize "hard" theories as those that have been extensively studied, confirmed, and which have contributed to a deep understanding of a given phenomenon. Studies to confirm such "hard" theories typically adopt strategies such as laboratory experiments which facilitate a high level of control and precision of measurement. Conducting field experiments in industry settings is inherently limited given that such natural settings do not facilitate the level of control that researchers would require to achieve a high level of precision of measurement, as would be possible in laboratory experiments [9]. By distinguishing hard theoretical from soft-theoretical approaches, this work aims to bridge the gap between researchers' goals and practitioners' possibilities.

In this paper we focus on quantitative research while we acknowledge the important role of qualitative research in nascent research areas. This focus allows us to elaborate and position 'soft theory' vs. 'hard theory.' The remainder of this paper is structured as follows. We discuss soft vs. hard theory

[1]We use the term 'theory' in the broadest sense.

in Sec. II. We adopt Leek's taxonomy describing different levels of data analysis in Sec. III to position soft and hard theoretical approaches. We conclude with some suggestions for future work in Sec. IV.

II. Soft vs. Hard Theory

In this paper we elaborate on the use of what we call soft theory, as a pragmatic alternative for conducting quantitative empirical studies in industry settings. Soft theory can be differentiated from hard theory along several dimensions (see Table I). Researchers who wish to confirm and extend already established theories work in a hard theory setting, while if their aim is to provide an initial, exploratory understanding of a new subject matter, soft theory is more appropriate. Soft theory approaches were initiated several decades ago in disciplines such as Management supported by advances made in statistics [10], where mathematical techniques for soft modeling are used to provide efficient predictions based on research, constrained by conditions of low information, nascent or emerging theory, and limited observations of a phenomenon [11]. There are many situations where a soft theory approach may be best suited for both scholars and practitioners, since it offers a pragmatic trade-off to those contexts where controlled experiments cannot be easily conducted, and the results of a soft theory approach can achieve *"good enough"* results that are useful to practitioners.

With a soft theory approach, the focus is not on questions such as *"which factors cause what effect,"* while trying to eliminate confounding factors that are very hard to rule out. Instead, soft theory approaches focus on correlation (e.g., path coefficients) and the proportion of the variance for a dependent variable that can be explained by an independent variable (i.e., R^2). In other words, these approaches highlight the level of significance of the different research hypotheses and explain how good a theoretical model is, compared to a zero-baseline model (which does not use any independent variables to predict the value of a dependent variable, so that it cannot say anything about the different relations). As a consequence, if research hypotheses are not firmly grounded in theory, the research model will have low path coefficient and R^2 coefficients, suggesting that the research model is purely random. Control over confounding factors is here purposely neglected for two reasons. Firstly, doing so is a more economical choice, because simpler research designs require less resources and effort to implement. Secondly, if those confounding factors are a constant within a specific industrial setting (and they cannot be discarded), they would not change the resulting soft theory anyway.

TABLE I
Comparison of soft theory vs. hard theory in SE research

	Soft Theory	Hard Theory
Context	The research domain is typically new or poorly studied from a knowledge-seeking perspective [9]	Well established research domain, where many insights have been gathered by previous and long lasting research
Goal	To provide introductory understanding of a software engineering phenomenon	To confirm and extend established software engineering theories
Focus	A preliminary understanding of a phenomenon which can not be explained with established theories	An already well-identified phenomenon; understanding causality
Outcome	Hypothesis, conceptual models, and correlation-based empirical evidence	Causation-based theoretical models
Example	The reasons episodic volunteers contribute in FLOSS communities is a topic of emergent interest to investigate alternative workforce in software development [12]	Software cost estimation theory is a very established theory. COCOMO was firstly proposed by Boehm et al. [13] in 1981 and still generates a literature debate to improve and extend the original model (e.g. COCOMO II [14])

III. Data Analysis in SE Research

We draw on Leek's framework which categorizes different types of data analysis for quantitative studies [15]. We discuss the different types according to the level of research effort that is usually required. A more intense research effort can lead to more precisely defined theories, but this may not always be possible. Besides the *cost* of doing research, hard theory studies cannot be performed in a reliable way in some cases. For example, if a phenomenon of interest is not well understood, measurement instruments may be lacking, which means that it is extremely challenging, if not impossible, to measure the concepts in an accurate manner.

There is a rich variety in empirical research methods, each with specific strengths and weaknesses [9]. In this paper, we focus exclusively on quantitative research and data analysis. Leek proposed a classification of data analysis approaches, which comprises six categories representing an increasing level of research effort: Descriptive, Exploratory, Inferential and Predictive, Causal, and Mechanistic [15]. The flowchart in Fig. 1 explains the level of depth of each category. In Table II we compare such types with examples. We discuss each type of data analysis below.

Descriptive analysis simply summarizes a data set without any interpretation. This is a typical initial step in most statistical analyses that can be used, for example, to assess distributional assumptions to decide whether one should use parametric or non-parametric approaches. This kind of analysis is useful to gain initial insights in big data sets, such as census data, without drawing any statistical conclusions.

Exploratory analysis looks for trends, outliers, correlations, and relationships of data derived from the descriptive analysis. Usually visualization approaches are used in this case. To continue the example of census data introduced above, specific patterns can be analyzed, such as: "for which reasons do we see a decrease of the birth rate with an increase of GDP/per capita?"

An *inferential* analysis quantifies whether a pattern holds beyond the data set, making no statements on individual situations. For example, a student might be interested to forecast

the *average* salary of a college graduate in five years time, at the point of graduation. To that end, data of college alumni may be used for the analysis, assuming that the new graduate will receive similar remuneration to that of current graduates. Thus, inference is an appropriate approach to look for average effects.

A *predictive* analysis focus on individual conditions. Using the example of before, to predict the individual 5-years salary of a college graduate a data analysis will use a range of factors about an individual, such as grades, family background, previous experiences, and so on. The results of this approach are not suitable for all graduates, but best suited for the analyzed individual. The difference between inferential and predictive analysis is whether or not the aim is to infer to a general population, or to predict an outcome for a particular individual.

Causal analysis typically requires randomized controlled experiments to test what happens *on average* to the dependent variable, after the treatment of the independent variable, in order to identify both direction and magnitude of the relationship. Directed acyclic graphs (DAG) are more often used to visualize causal, confounding, and bias of analyzed phenomena [16].

Finally, a *mechanistic* analysis seeks very precise, deterministic behaviors on such relationships. Taking the medical domain as a typical example: a very precise, mechanistic analysis can help to determine the exact amount of medication to prescribe to patients with a specific health issue. To establish this level of precision, studies have to be modeled by a deterministic set of equations that consider a range of relevant variables such as age, gender, and variables related to health conditions (e.g. blood pressure). The deterministic set of equations, then, model the ranges of values for these variables. A mechanistic analysis provides more information than a "binary" outcome indicating whether or not a specific drug works, as one would find in a typical A/B test setting.

According to Fig. 1, descriptive analysis is not considered a theory-generating approach, because it does not draw any statistical conclusions (cf. [2]). On the other hand, with different degrees of research accuracy, exploratory, inferential, and predictive analysis can be fruitfully used to support soft theory models, clearly stating the limitations inherent to the selected method.

Fig. 1 provides an overview of the approaches with increasing research effort. As mentioned in the introduction, in many cases a soft theory approach may be suitable for industry research contexts, especially if they focus on inference and prediction. We must acknowledge that the difference between inferential and predictive approaches, according to Leek's framework is very blurred in software engineering. Software engineering research rarely focuses on *individual* results but usually seeks generalizability. However, this distinction leads to a reflection on conclusive claims that we are used to make related to the data we use. If we are looking at one specific organization, and are conducting solution-seeking research [9] that leads to new tools or models which are best tailored to that organization, it makes very little sense to claim for average effects. Such results may still be useful because that particular solution is likely to lead to improvements for that specific case, rather than a generic average-based one.

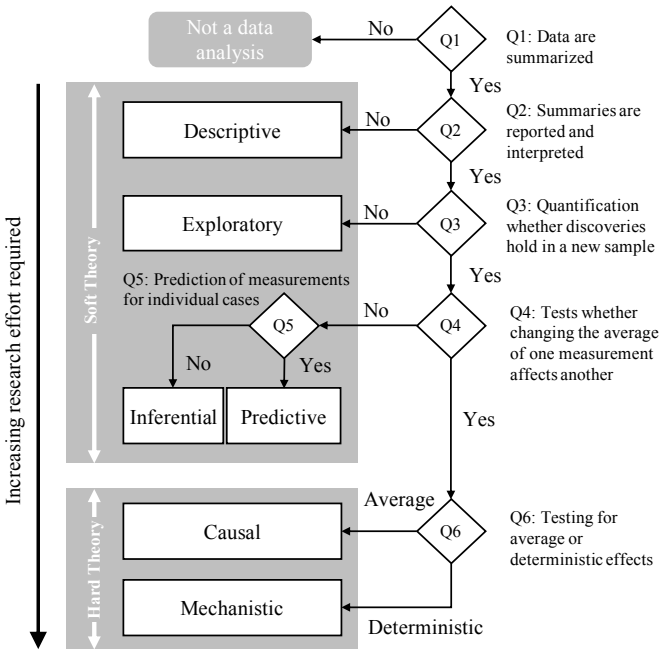

Fig. 1. Hierarchical Data Analysis Flowchart for Software Engineering research (adapted from Leek [15])

We acknowledge the increasing attention to Bayesian Data Analysis in SE research [17]. In this context, we see BDA as a valuable approach for hard theory cases, since conditional probability is gathered by previous studies and consolidated experiences, and this conflicts with our premise that in nascent areas such prior data and studies are lacking.

IV. Conclusion

Researchers in other fields such as Management have adopted data analysis approaches, which we have labeled "soft theory" approaches, to overcome the typical constraints of *in-vitro* settings that only laboratory settings can offer. In other words, it provides a snapshot of a phenomenon, without the possibility to manipulate variables, when it is not possible to test for causation through randomized controlled experiments. Hence, industry constructs for which we lack a thorough understanding can be tested and validated using soft modeling approaches.

This paper proposes the use of soft theory as a pragmatic alternative for conducting industry studies in the software engineering domain. We position soft modeling techniques, such as Partial Least Squares Structural Equation Modeling (PLS-SEM), in Leek's framework (see Fig. 1), and contrast it with what we have term hard theory (see Table I). Leek's framework positions a range of data analysis techniques (see Table II), most of which are suitable for SE research. Soft theory is a convenient initial approach to work towards hard theory, especially in early stages of research on new and emerging phenomena, where the inferential and predictive potentials of statistical methods can provide statistically rigorous outcomes.

TABLE II
DESCRIPTION OF DATA ANALYSIS TYPES IN SOFTWARE ENGINEERING BASED ON LEEK'S TAXONOMY [15]

Data Analysis	Aim	Typical methods	Common mistakes	Example	Effort
Descriptive	Description of a data set. Useful to have a first understanding of the phenomenon	Univariate analysis (e.g. histograms, pie charts, etc.)	Drawing conclusions beyond the data set description	Industry report on adoption of Agile practices by VersionOne [18]	Very low
Exploratory	Find previously unknown relationships. Useful to generate research hypotheses	Correlation analysis	Data "fishing": looking for patterns and other information in a data set that it does not actually contain	The analysis of a data set on object-oriented metrics suggest that they are significantly correlated with software management indicators [19]	Low
Inferential	Test hypotheses about a general phenomenon. Computations are based on generic datasets (e.g., software repositories)	Regression models, path modeling (e.g., PLS-SEM)	Claiming for causation	On average, post-release defects decreases with code review coverage, participation, and expertise of developers [20]	Medium
Predictive	Test hypotheses about an individual phenomenon. Computations are based on individual historical data (e.g., time series)	Regression models, path modeling (e.g., PLS-SEM)	Used for generalization purposes	Framing of defect prediction models using metrics for one organization (NASA), extracted from the NASA datasets [21]	Medium
Causal	Finding causal relations between different phenomena	Randomized Controlled Experiment (RCE), confirmatory research (e.g. CB-SEM), Bayesian Data Analysis (BDA)	Make deterministic claims on causal relations	Tool supports lead to significant improvements of quality metrics but does not affect the number of bugs found by developers [22]	High
Mechanistic	Determine exact changes on independent variable to cause an exact reaction on the dependent one	RCEs modeled by a deterministic set of equations, BDA	Not applicable, because mechanistic analysis is the most fine-grained type of analysis available.	The aim of COCOMO is to compute the software development effort as function of program size and a set of cost drivers [14]	Very high

Future work will focus on relatively unknown inferential-predictive approaches in software engineering, in particular PLS-SEM, providing clear methodological guidelines for software engineering researchers.

ACKNOWLEDGMENTS

This work was supported with the financial support of the Science Foundation Ireland grant 15/SIRG/3293 and 13/RC/2094 to Lero—the Irish Software Research Centre.

REFERENCES

[1] P. Johnson, M. Ekstedt, and I. Jacobson, "Where's the theory for software engineering?" *IEEE Software*, vol. 29, no. 5, pp. 96–96, 2012.
[2] K.-J. Stol and B. Fitzgerald, "Theory-oriented software engineering," *Science of Computer Programming*, vol. 101, 2015.
[3] M. Hamilton, "Keynote at 50th International Conference on Software Engineering," https://www.youtube.com/watch?v=ZbVOF0Uk5lU, May 2018.
[4] M. Conway, "How do committees invent," *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.
[5] F. P. Brooks, *The mythical man-month. Essays on software engineering*. Addison-Wesley, 1982.
[6] A. Davis, *201 Principles of software development*. McGraw-Hill, 1995.
[7] C. Alexander, *A pattern language: towns, buildings, construction*. Oxford University Press, 1977.
[8] J. Hannay, D. Sjoberg, and T. Dyba, "A systematic review of theory use in software engineering experiments," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 87–107, 2007.
[9] K.-J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Trans Softw Eng Methodol*, vol. 27, no. 3, 2018.
[10] H. Wold, "Systems analysis by partial least squares," in *Measuring the unmeasurable*, P. Nijkamp, H. Leitner, and N. Wrigley, Eds. Dordrecht, 1985, pp. 221–252.
[11] J. Sosik, S. Kahai, and M. Piovoso, "Silver bullet or voodoo statistics? a primer for using the partial least squares data analytic technique in group and organization research," *Group & Organization Management*, vol. 34, no. 1, pp. 5–36, 2009.
[12] A. Barcomb, K.-J. Stol, D. Riehle, and B. Fitzgerald, "Why do episodic volunteers stay in floss communities?" in *Proceedings of the 41th International Conference on Software Engineering*. ACM/IEEE, 2019.
[13] B. W. Boehm *et al.*, *Software engineering economics*. Prentice Hall, 1981.
[14] ——, *Software cost estimation with Cocomo II with Cdrom*. Prentice Hall, 2000.
[15] J. Leek and R. Peng, "What is the question?" *Science*, vol. 347, no. 6228, pp. 1314–1315, 2015.
[16] T. Williams *et al.*, "Directed acyclic graphs: a tool for causal studies in paediatrics," *Pediatric Research*, vol. 84, no. 4, pp. 487–493, 2018.
[17] C. Furia, R. Feldt, and R. Torkar, "Bayesian data analysis in empirical software engineering research," *arXiv preprint arXiv:1811.05422*, 2018.
[18] VersionOne, "13rd annual state of agile survey," 2018. [Online]. Available: http://stateofagile.versionone.com/.
[19] S. Chidamber, . Darcy, and C. Kemerer, "Managerial use of metrics for object-oriented software: An exploratory analysis," *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 629–639, 1998.
[20] S. McIntosh, Y. Kamei, B. Adams, and A. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2146–2189, 2016.
[21] Y. Jiang, B. Cukic, and T. Menzies, "Fault prediction using early lifecycle data," in *Proceedings of the 18th IEEE International Symposium on Software Reliability*. IEEE, 2007, pp. 237–246.
[22] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated unit test generation really help software testers? A controlled empirical study," *ACM Trans Softw Eng Methodol*, vol. 24, no. 4, 2015.