Theory, models and methods in software engineering research.

Ross Jeffery

Centre for Advanced Empirical Software Research, School of Information Systems Technology and Management, University of New South Wales, Kensington, 2052 NSW, Australia
+61.2.93854735
r.jeffery@unsw.edu.au

Abstract

This paper uses examples of research in software engineering carried out in the Centre for Advanced Empirical Software Research to show the manner in which theory from other disciples such as the behavioral sciences can be used as a basis for hypothesis formulation. It also investigates the process of matching the appropriate research methods with the goals and available theory and models.

Theory and Models

Developing appropriate theory for empirical testing in software engineering is a challenge to the empiricist. On occasion such models and theory may be available in the behavioral and cognitive sciences. At other times the appropriate theory and models may be based in computer science or measurement theory. Developing the appropriate linkages provides opportunity to expand our research horizons and to add significantly to the body of knowledge in software engineering. In this paper examples of these linkage are provided and suggestions made that guide the application of research paradigms.

In this paper we consider an area of research in which The Centre for Advanced Empirical Software Research (CAESAR) has been working as an example. Work into software development technical reviews (SDTR's) has been based on laboratory experiment [8], case study [6], and survey [6]. The experiments were conducted in a classical scientific research mode in which theory development [3] preceded the development of research questions and hypotheses. The work done showed that the behavioral theory of group performance was applicable to the software review task. We then developed a program of empirical research based on that behavioral theory, including propositions to (1) explain review performance, and (2) identify ways of improving review performance based on the specific strengths of individuals and groups. It made three contributions: (1) it clarified our understanding of what drives defect detection performance in technical reviews; (2) it set an agenda for future research; and (3) it demonstrated the value of a theory-driven research program. In identifying individuals' task expertise as the primary driver of review performance, the research program suggested specific points of leverage for substantially improving review performance and pointed to the importance of understanding software reading expertise and implied the need for a reconsideration of existing approaches to managing reviews.

In this instance, the characteristics of the state of practice in research indicated that there was theory available in the behavioral sciences that could be usefully applied in the software engineering domain. The theory had been empirically tested in other decision-making and review settings, thus strengthening the likelihood of successful adoption in our domain. Extensive work in theory adoption and model building for the domain of interest resulted in a series of justifiable research propositions.

The propositions advanced were:

P1 In SDTRs, task expertise is the dominant determinant of group performance
P2 In SDTRs, decision schemes (plurality effects) determine interacting group performance
P3 In SDTRs, in the absence of a plurality, interacting group performance is a positive function of process skills
P4 In SDTRs, the interacting group meeting does not improve group performance over the nominal group by discovering new defects
P5 In SDTRs, task training leads to improved group performance
P6 In SDTRs, the performance/size relationship is a function of task expertise
P7 In SDTRs, above a critical limit, performance declines with size
P8 In SDTRs, the performance advantage of an interacting group over a nominal group is a function of the level of false positives discovered by individuals
P9 In SDTRs, an expert pair performs the discrimination task as well as any larger group
P10 In SDTRs, nominal groups outperform alternatives at the discovery task
P11 In SDTRs, the defect discovery performance/size relationship for nominal groups is a function of task expertise

The focus provided by the underlying theory makes it natural to pursue cumulative research. Using behavioral theory to predict determinants of review performance has led us to distinguish the three tasks of defect discovery, collection and discrimination. Figure 1 illustrates how each proposition in the program contributes to our understanding of one of these stages. Research on multiple propositions relating to a task will help build a better understanding of that task. Research on all three tasks will lead us to an understanding from which we can develop and tune SDTR designs.

One of the strengths of the explanatory capacity of the underlying theory is the insight it gives us beyond the set of propositions formally constituting the program. For example, behavioral theory allows us to see that task-oriented meeting roles [12], review specialization [7], [[9] and the scenario task aid [10] all work on the same principle, namely that defect detection is enhanced by specialization in the application of expertise. This immediately leads us to ask whether the source of any improvement is the need for different expertise or something quite different such as the reduction in size of the individual tasks. The answer implies quite different interventions.

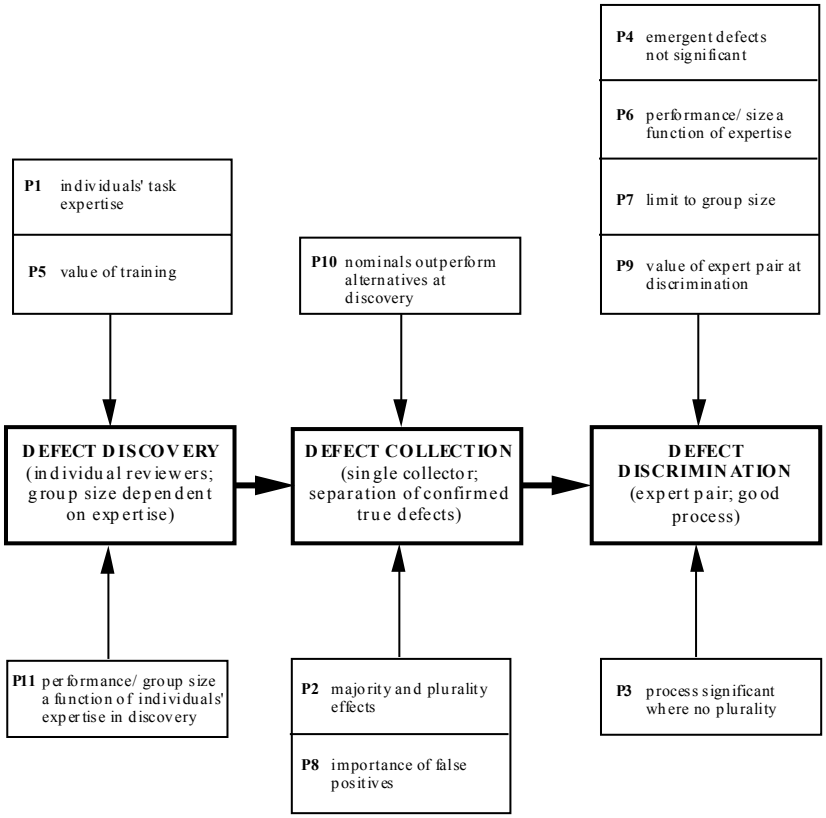

Figure 1: Relationship of research propositions to SDTR component tasks

Lacking a theory of defect detection expertise, the program highlights the urgency of addressing this lack.

Inductive Models

In this example we have a classical research approach in which theory could be utilized to design the research program and then appropriate research methods used to address aspects of that program. This example contrasts with the situation in which we found ourselves on the CADPRO project (Constraints and the Decision Process) [4]. In this instance we were investigating the impact of constraint variation in CASE tools on the early lifecycle productivity of designers and the quality of the design produced. In this instance we were not able to borrow theory but instead were able to develop a research model from self-reported survey data collected from software designers. In essence we set about to develop a research model which following empirical; testing might eventually expose theory. The model used is shown in Figure 2. (from [2]) In this instance the research model was developed from a single survey. As such it would be expected that the model would be likely to need change based on further empirical and theoretical study. One example of this model development is provided in [11] who modified this research model based on the writings of [1] adding additional cognition elements to the model.

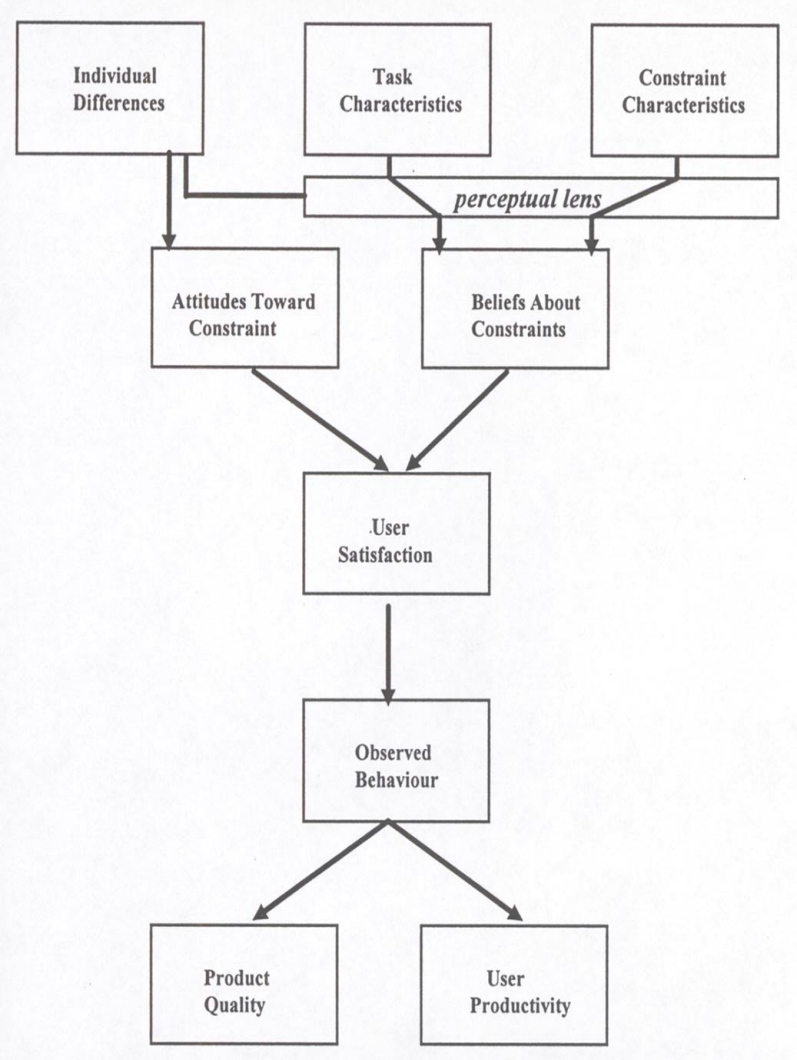

Figure 2. CADPRO Initial Research Model

In both the CADPRO and SDTR research we were able to "borrow" elements of the research models from behavioral and cognitive research literature. This opportunity is not always afforded to the software engineering researcher.

Fitness for Purpose

In another recent project CAESAR researchers investigated the application of Full Function Points (FFP) as a measure of system size to the application domain of embedded telephone switching systems [13]. In this case the research method employed was a case study and the goals weary to investigate the impact on measured system size of the dual issues of (1) definition of the counting boundary and (2) level of abstraction. In this case there is considerable literature in software metrics concerning Function Points (FPA) and almost none concerning FFP. The choice of research method was defined by the research context. The research was carried out for a large multi-national telecommunications company whose interest was in the application of the counting technique to their software process. Since they had not used the technique currently or previously, research methods such as field study or survey were not possible. Here we see an example of software engineering research that seeks to answer some typical technical questions concerning the software product. Although there is a very large literature on software metrics, there is little applicable theory for the question of metric suitability to this particular context. We had no interest in the theoretical basis of the metric, but in its practical application or "fitness for purpose".

Discussion

Indeed a central issue that needs to be addressed by the research community is:

1. When is the research issue one for empirical/scientific research, and when is it better treated by other forms of research?
2. If it is an issue of empirical/scientific research, what advantages and disadvantages accrue from the methods available?

In the first case, if the researcher produces a new method or tool, then perhaps it will be appropriate to defer to the market evaluation of the contribution. For example the FPA sizing method has been shown to be theoretically deficient in its structure [5]. However no alternate sizing method has been adopted widely in international practice and hence the market makes the judgement that the method is sufficiently "fit for purpose" to warrant its use in practice. In this type of case, we might be able to make valuable contribution to practice by providing insight into that state of practice or else we might be better putting effort into developing new methods for later industrial trial. This is, of course, the more traditional domain of software engineering; that of developing methods and tools.

In the second case, where the research questions are clear and the answer to those questions lies in empirical methods, then the contribution should be made in the development of theory and models, and the methods that can be best used to test the hypotheses that derive from those theories and models.

References

[1] R.Agarwal, A.P.Sinha and M.Tanniru, "The role of prior experience and task characteristics in object oriented modeling: an empirical study", *Int. J. Human-Computer Studies*, 45, pp. 639-667, 1996.

[2] D. Day, "Behavioral and perceptual responses to the constraints of computer-mediated design", In Brouwer-Janse, M. & Harrington, T. (Eds.), *Human-Machine Communication for Educational Systems Design.* Series F: Computer and Systems Sciences Vol. 129, NATO ASI Series. Berlin: Springer-Verlag, 1994.

[3] C. Sauer, R. Jeffery, L. Land, P. Yetton, Understanding and Improving the Effectiveness of Software Development Technical Reviews: A Behaviourally Motivated Programme of Research, to *IEEE Transactions on Software Engineering*, Jan/Feb, 2000.

[4] D. R. Jeffery and R. J. Offen (Eds.) *Experiments in CASE Tool Use and Constraint Conditions*, CAESAR/University of New South Wales and JRCASE/Macquarie University, 25th September 1999, 95pp.

[5] D. R. Jeffery and J. Stathis, "Function Point Sizing: Structure, Validity and Applicability", Empirical Software Engineering: An International Journal, Kluwer Academic Publishers, Boston, 1996, pp 11-30.

[6] Kim, L. P. W., Sauer, C., Jeffery, R. 1995. A framework of software development technical

reviews, Software Quality and Productivity: Theory, Practice, Education and Training, (Eds.) Matthew Lee, Ben-Zion Barta, Peter Juliff, Chapman and Hall, 294-299, IFIP.

[7] J.C. Knight and A.N. Myers, "An improved inspection technique", *Communications of the ACM*, vol. 36, no. 11, pp. 51-61, November 1993.

[8] Land, L. P. W., Sauer, C., Jeffery, R. 1997a. Validating the defect detection performance advantage of group designs for software reviews: Report of a laboratory experiment using program code, Proceedings of the 6 th European Software Engineering Conference held jointly with the 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Zurich, Switzerland, 22-25 September, Lecture Notes in Computer Science 1301, (Eds.) Goos, G., Leeuwen J.V., 294-309,Springer-Verlag.

[9] D. L. Parnas and D. M. Weiss, "Active design reviews: principles and practices," in *Proc. 8th International Conference on Software Engineering*, IEEE & ACM, August, 1985, pp. 215 - 222.

[10] A.A. Porter and L.G. Votta, "An experiment to assess different defect detection methods for software requirements inspections," in *Proc. 16th International Conference on Software Engineering*, IEEE & ACM, Sorrento, Italy, May 16-21, 1994, pp. 103-112.

[11] E. Robertsson, H. Eriksson. *An Empirical Study on Product and Process Quality in Object-Oriented Design.* CAESAR Technical Report, Centre for Advanced Empirical Software Research, Tech00-8, 2000, 181 pp.

[12] E. Yourdon, *Structured Walkthroughs*, 4th edition, Prentice-Hall, 1989.

[13] Suryaningsih, The applicability of function points to the domain of embedded telephone switching systems, CAESAR Technical Report 00/10, Centre for Advanced Empirical Software Research, www.caesar.unsw.edu.au, 2000.