Empirical Software Engineering
https://doi.org/10.1007/s10664-018-9655-0

VIEWPOINT

Four commentaries on the use of students and professionals in empirical software engineering experiments

Robert Feldt[1] · Thomas Zimmermann[2] · Gunnar R. Bergersen[3] · Davide Falessi[4] · Andreas Jedlitschka[5] · Natalia Juristo[6] · Jürgen Münch[7] · Markku Oivo[8] · Per Runeson[9] · Martin Shepperd[10] · Dag I. K. Sjøberg[3,11] · Burak Turhan[12]

Published online: 21 November 2018

The relative pros and cons of using students or practitioners in experiments in empirical software engineering have been discussed for a long time and continue to be an important topic. Following the recent publication of "Empirical software engineering experts on the use of students and professionals in experiments" by Falessi, Juristo, Wohlin, Turhan, Münch, Jedlitschka, and Oivo (EMSE, February 2018) we received a commentary by Sjøberg and Bergersen. Given that the topic is of great methodological interest to the community and requires nuanced treatment, we invited two editorial board members, Martin Shepperd and Per Runeson, respectively, to provide additional views.

✉ Robert Feldt
robert.feldt@chalmers.se

[1] Chalmers University of Technology and Blekinge Institute of Technology, Gothenburg, Sweden
[2] Microsoft Research, Redmond, WA, USA
[3] Department of Informatics, University of Oslo, Oslo, Norway
[4] California Polytechnic State University, San Luis Obispo, CA, USA
[5] Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
[6] Universidad Politécnica de Madrid, Madrid, Spain
[7] Reutlingen University, Reutlingen, Germany
[8] University of Oulu, Oulu, Finland
[9] Department of Computer Science, Lund University, Lund, Sweden
[10] Brunel University, London, UK
[11] SINTEF Digital, Trondheim, Norway
[12] Faculty of Information Technology, Monash University, VIC, Australia

Finally, we asked the authors of the original paper to respond to the three commentaries. Below you will find the result. Even though we are under no illusion that these views settle the issue we hope you find them interesting and illuminating, and that they can help the empirical software engineering community navigate some of the subtleties involved when selecting representable samples of human subjects.

– Robert Feldt and Thomas Zimmermann, Editors-in-Chief of Empirical Software Engineering Journal.

The Price of Using Students

Dag I.K. Sjøberg and Gunnar R. Bergersen

In a recent article, Falessi et al. (2018) call for a deeper understanding of the pros and cons of using students and professionals in experiments. The authors state: "we have observed too many times that our papers were rejected because we used students as subjects."

Good experiments with students are certainly a valuable asset in the body of research in software engineering. Papers should thus not be rejected solely on the ground that the subjects are students. However, the distribution in skill is different for students and professionals. Since previous studies have shown that skill may have a moderating effect on the treatment of participants, we are concerned that studies involving developers with only low to medium skill (i.e., students) may result in wrong inferences about which technology, method or tool is better in the software industry. We therefore provide suggestions for how experiments with students can be improved and also comment on some of the alleged drawbacks of using professionals that Falessi et al. point out.

1 Different Distributions for Students and Professionals

While we know that the level of skill among subpopulations of students varies substantially, less attention has been paid to the variation in skill among subpopulations of professionals, which is also substantial. For example, we previously found that the benefit of using pair programming was positive for juniors, whereas it was negative for senior developers (Arisholm et al. 2007). In another experiment, only the best performing developers benefitted from a "proper" object-oriented control style, whereas the others did not (Arisholm and Sjøberg 2004). Also, in a study by Krein et al. (2016), the purported benefit of the investigated design patterns was mostly negative for the students but mostly positive for the most experienced and knowledgeable professionals. Such experiments show that one technology (method, technique, tool, etc.) may be better for developers at one skill level, while another technology may be better for developers at another skill level, as illustrated in Fig. 1. This "reversal effect" (Sjøberg et al. 2016) needs to be taken into account when designing and analyzing experiments.

Another reason for the apparently contradictory results in the literature on the differences between students and professionals is that the skill distribution of various subpopulations is rarely taken into account. Figure 2 illustrates the distributions of five categories of developers. The graphs representing junior, intermediate and senior professionals are based on aggregated data from three experiments ($n = 262$) lasting one or two days (Arisholm and Sjøberg 2004, Arisholm et al. 2007, Bergersen et al. 2014).

The exact form of the curves of the undergraduate and graduate student populations are hypothesized because we do not have comprehensive data from multiple experiments that include both students and professionals, but the mean skill of students is placed to the left of the mean skill of professionals (i.e., less skilled) based on the well established *theoretical consideration* that people improve their skills through practice, cf. the law of practice (Newell and Rosenbloom 1981). Note that this theoretical consideration is based on hundreds of empirical studies in various disciplines. If one thinks in terms of *cohorts*, developers undergo

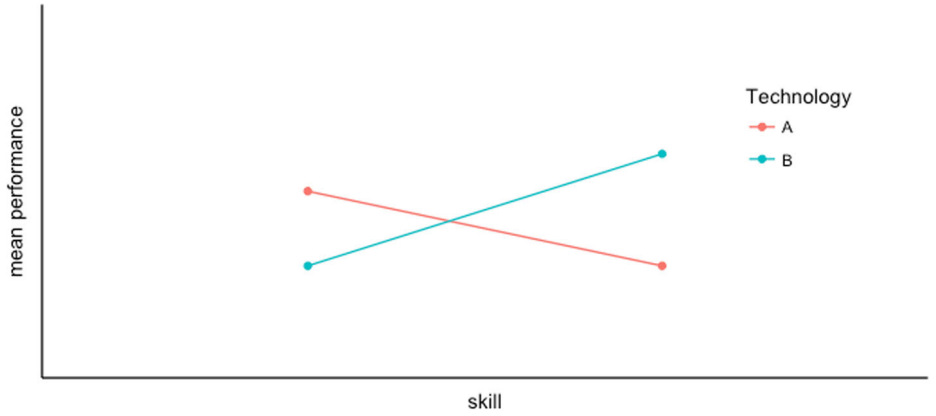

Fig. 1 Which technology is best depends on skill level

transitions from undergraduate to graduate students and then to junior, intermediate, and senior professionals, working with increasingly greater skill throughout their education and their professional career.

In practice, however, many developers do not transition through all the categories. For example, some undergraduates start working as juniors after their BSc degree; some companies do not have an "intermediate" professional developer category; some graduates may advance directly into the intermediate category; and so on. In addition, while published experiments show that the mean skill of undergraduates is generally lower than that of professionals, the mean skill of graduate students might be higher than that of junior professionals, depending on the concrete topic of investigation and domain of skill. In any case, the overall distribution of students will be to the left of the overall distribution of professionals (see the shaded areas). Consequently, one should be cautious about generalizing when the sample is drawn from a population (students) other than that which one aims to generalize to (professionals).

Fig. 2 Distribution of skill in different populations

For any specific and well-defined skill, individual differences can be substantial. Moreover, different skills will be distributed differently, and which skill is relevant in a given experiment will depend on the experimental treatment. For example, if the technology being evaluated in an experiment recently had been taught to a group of students, they may be *more* skilled in that technology than a group of professionals who had not recently used that technology. Such a case supports the argument of Falessi et al. that it is not always better to experiment with professionals.

The sample size of most studies in software engineering is small and subjects are sampled by convenience. The samples are therefore rarely representative of the underlying student or professional populations. Highly skilled students and less skilled professionals will be sampled on occasion. A study may then easily show that students perform better than or similar to professionals. Similar results between students and professionals were obtained by, for example, Höst et al. (2000), which might have been surprising at the time but may be expected given non-random sampling from overlapping distributions (Fig. 2). Moreover, the study contrasted (subjective) judgments of students versus professionals and not (objective) performance-based measures where skill is more central. Thus, we caution against using a few (outlier) studies to justify that students are good proxies for professionals in general or in a given study.

Falessi et al. state that "many more comparative studies are needed before we obtain an answer on whether students are good proxies of professionals in software engineering experiments." Since students and professionals have different characteristics, one can hardly ever claim that students are good proxies for professionals *in general*, but certain student samples may be good proxies for certain subsets of professionals. Careful empirical investigation is indeed needed to detect those cases.

2 Alleged Drawbacks of Professionals

Falessi et al. conducted a survey among empirical researchers and reported that the respondents in general disagreed with Falessi et al. about the drawbacks of professionals. We also have several objections to the drawbacks stated by Falessi et al. First, they claim that paying professionals for participating in experiments is a "strong threat to validity." The Merriam-Webster dictionary defines "professional" work as being "engaged in by persons receiving financial return." Professionals are thus by definition paid for their work, whether for work done in experimental or non-experimental settings. The money is not additional payment to individuals to work extra in their free time, but rather compensates the organization for those who participate in a study during ordinary (paid) work hours. A sample may easily be biased if only those who are willing to spend their spare time are included in the sample because such volunteers may possess characteristics that affect the experimental outcome that are different from the characteristics of the typical paid industrial developer. In the same vein, students may participate in experiments as part of mandatory coursework or may be rewarded by extra course credits. In our view, compensation for participation in experiments motivates developers in a way similar to that of ordinary work, thus reducing the likelihood of confounding effects of differences in motivation between the available sample and the target population.

Second, Falessi et al. state that there is a tradeoff between higher *internal* validity when using students versus higher *external* validity when using professionals. Clearly, there are fewer concerns about external validity in experiments with professionals, but the moderating

effect that skill may have on the treatment of an experiment (see Section 1) is a particular threat to the internal validity of student-based experiments. Moreover, it is certainly possible to have better internal validity in a well-designed student study than in a poorly designed professional study, but that does not mean that student experiments have better internal validity per se. To support the claim that internal validity is lower when using professionals, Falessi et al. refer to a book on research methods in health sciences, which states that "the relationship between internal and external validity is inverse" (Berg and Latin 2003, p. 213). According to this premise, any student experiment would have higher internal validity than any experiment with professionals, which is clearly not the case. Software engineering is not a typical lab science; the same kind of treatment and experimental control can be applied to professionals as well as to students. The level of internal validity may be the same in a student experiment as in an experiment with professionals, but the latter may have higher external validity.

Third, another drawback to using professionals that Falessi et al. claim is that sample sizes are small and that professionals tend not to show up for experiments. Without any budget for recruiting subjects, it is certainly easier to obtain larger sample sizes with students. It is easier to obtain many mice for a medical experiment, but medical researchers do not consider using humans to be a drawback. Moreover, if one follows a well-designed procedure for recruiting professionals, including contracted payment with the organizations of the professionals and conduct the experiments in regular work hours (Sjøberg et al. 2007), they will show up. In our own experiments, the no-shows are negligible and comparable to the percentage of individuals that would be on sick leave for any given day.

Fourth, Falessi et al. claim that professionals are less committed than students to finish tasks on time because students "are used to an examination culture." Finishing tasks on time and keeping deadlines are certainly important in work life, and this claim is in stark contrast to what was the case in our experiments with hundreds of professionals.

3 Characterizing Subjects

We agree with Falessi et al. that our research field must characterize subjects beyond whether they are students or professionals. Based on a suggestion from one of the respondents in their study, Falessi et al. propose that subjects should be interviewed and qualitatively described using a characterization of real, relevant and recent experience (R^3). However, they define real experience as "experience judged relevant by the researcher in dialog with each subject," indicating that there is no difference between real and relevant experience in their definition. Nevertheless, there is empirical support for using recent experience (e.g., Sigmund et al. 2014), in particular when there is a good match in specificity between the experience predictor and the criterion being predicted (Bergersen et al. 2014). For example, if a study requires Java programming, "*Java* programming experience" will be a better predictor than "*general* programming experience."

Additionally, Falessi et al. suggest replacing a student-professional dichotomy with an experience trichotomy (0–2, 3–5, 5+ years of experience). Discretizing a continuous variable such as experience reduces statistical power and is, thus, a threat to statistical conclusion validity (Shadish et al. 2002). Furthermore, using years of experience may work well for professionals but not for students, who mostly have none. Because variables with little variance or with ceiling/floor effects reduce statistical power (Shadish et al. 2002), the experience of students should be measured in months.

Even though experience plays an important role in both theories of skill (Fitts and Posner 1967) and expertise (Ericsson and Charness 1994), it is knowledge, skill and motivation that cause subjects to perform well (Campbell et al. 1993). Thus, experience is a proxy for skill, which together with a long list of other variables can be used to predict how well a programmer will perform in an experiment (see Bergersen et al. 2014). Unfortunately, although experience has a medium positive correlation with performance (around 0.30 across our own studies), it is neither the best nor the only proxy one can use (one should consider, e.g., LOC and self-reported skills).

A better alternative to using proxy variables is to develop and use pretests. They enable powerful designs such as matching, stratifying or blocking (Shadish et al. 2002). An example of the use of a small, single-task pretest can be found in (Arisholm and Sjøberg 2004). The main experiment consisted of a set of change tasks on a Java program that was provided in two variants of control style (the treatment of the experiment). Before starting on the experiment tasks, all participants performed the same change task on another Java program (with no variants). Individual differences in performance on this pretest task were used in the analysis of the experimental task to adjust for differences between the treatment groups. An example of the use of a more comprehensive pretest is described in (Bergersen and Sjøberg 2012).

Using posttests may also support the investigation of several potential threats to validity (for details, see Kampenes et al. 2009). For example, we have used pre- and posttests to investigate the effects of learning (practice) during an experiment, which may undermine the claim that subjects are already highly skilled (often called "experts") on the technology they are working on (Sheil 1981). In a sample of 65 professional developers, the level of performance of subjects was stable after a small warm-up period, thus demonstrating that the professionals did not learn anything during the experiment (Bergersen et al. 2014).

An alternative to developing one's own pre- and posttests is to use validated instruments that previously have been shown to highly correlate with the dependent variable in an experiment and where the skill level of different categories of professional developers are available. We have developed such an instrument to predict programming performance (Bergersen et al. 2014). Admittedly, there are few such instruments available, and they require much effort to develop.

4 Conclusion

Overall, we are concerned that the low proportion of professionals as participants reduces the impact of software engineering experiments on industry. It is difficult to influence development practices by arguing that a group of students benefitted from using a certain method or tool when it is unknown how these students differ in skill and motivation relative to professional programmers. Other things being equal, sampling from the same population that one aims to generalize to reduces threats to validity. In an earlier literature review, we found that only 9% of the subjects were professionals (Sjøberg et al. 2005). More recently, we examined the experiments published in the journals *IEEE Transactions on Software Engineering* and *ACM Transactions on Software Engineering and Methodology* in 2015 and 2016, and *Empirical Software Engineering* in 2016 and 2017. Among the total of 1752 subjects, only 139 were professionals, that is, 8%. Consequently, we urge the community to run more experiments with professionals.

Nevertheless, it may be impractical and expensive to obtain appropriate samples of professionals. Using students is then a good alternative, although the number and magnitude

of potential threats to validity increase. Researchers should, therefore, carefully and cautiously report and discuss the limitations of student-based experiments. In particular, the moderating effect of skill level on the benefit of treatment should be addressed.

References

Arisholm E, Sjøberg DIK (2004) Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. IEEE Trans Softw Eng 30(8):521–534.

Arisholm E, Gallis H, Dybå T, Sjøberg DIK (2007) Evaluating pair programming with respect to system complexity and programmer expertise. IEEE Trans Softw Eng 33(2):65–86.

Berg KE, Latin RW (2003) Essentials of research methods in health, physical education, exercise science, and recreation. Lippincott Williams & Wilkins.

Bergersen GR, Sjøberg DIK (2012) Evaluating methods and technologies in software engineering with respect to developer's skill level. IET-EASE, pp. 101–110.

Bergersen GR, Sjøberg DIK, Dybå T (2014) Construction and validation of an instrument for measuring programing skill. IEEE Trans Softw Eng 40(12):1163–1184.

Campbell JP, McCloy RA, Oppler SH, Sager CE (1993) A theory of performance. In: Schmitt N, Borman WC (eds) Personnel selection in organizations. Jossey-Bass, San Francisco, CA, pp 35–70.

Ericsson KA, Charness N (1994) Expert performance: Its structure and acquisition. Am Psychol 49(8):725–747.

Falessi D, Juristo N, Wohlin C, Turhan B, Münch J, Jedlitschka A, Oivo M (2018) Empirical software engineering experts on the use of students and professionals in experiments. Empir Softw E. https://doi.org/10.1007/s10664-017-9523-3.

Fitts PM, Posner MI (1967) Human performance. Brooks/Cole, Belmont, CA.

Höst M, Regnell B, Wohlin C (2000) Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. Empir Softw Eng 5(3):201–214.

Kampenes VB, Dybå T, Hannay JE, Sjøberg DIK (2009) A systematic review of quasi-experiments in software engineering. Inf Softw Technol 51(1):71–82

Krein JL, Prechelt L, Juristo N, Nanthaamornphong A, Carver JC, Vegas S, Knutson CD, Seppi KD, Eggett DL (2016) A multi-site joint replication of a design patterns experiment using moderator variables to generalize across contexts. IEEE Trans Softw Eng 42(4):302–321.

Newell A, Rosenbloom P (1981) Mechanisms of skill acquisition and the law of practice. In: Anderson JR (ed) Cognitive skills and their acquisition. Erlbaum, Hillsdale, NJ, pp 1–56.

Shadish WR, Cook TD, Campbell DT (2002) Experimental and quasi-experimental designs for generalized causal inference. Houghton Mifflin, Boston.

Sheil BA (1981) The psychological study of programming. ACM Comput Surv 13(1):101–120.

Siegmund J, Kästner C, Liebig J, Apel S, Hanenberg S (2014) Measuring and modeling programming experience. Empir Softw Eng 19(5):1299–1334.

Sjøberg DIK, Bergersen GR, Dybå T (2016) Why theory matters. In: Williams L, Menzies T, Zimmermann T (eds) Perspectives on data science for software engineering. Morgan Kaufman, Amsterdam, pp 29–32.

Sjøberg DIK, Hannay JE, Hansen O, Kampenes VB, Karahasanovic A, Liborg N-K, Rekdal AC (2005) A survey of controlled experiments in software engineering. IEEE Trans Softw Eng 31(9):733–753.

Sjøberg DIK, Dybå T, Jørgensen M (2007) The future of empirical methods in software engineering research. In: Future of Software Engineering. IEEE-CS, pp. 358–378.

Realism in Software Engineering Experiments – A Search in Vain?

Per Runeson

1 Introduction

During my PhD studies in the late 1990's, I conducted experiments on software inspections. The overall goal was to estimate the remaining defect content after an inspection. We launched a quasi-experiment with students and professionals to get data for the estimation studies and – as a by-product – we observed a correlation between the subject's experience and the review effectiveness. However, in contrast to our expectations, the correlation was *negative*, i.e. more experienced reviewers found less faults (Runeson and Wohlin 1998).

This observation lead me first into double and triple checking my calculations, and then into rethinking the notion of experience, whether the number of working years is a feasible predictor of inspection effectiveness. It gave me an insight – perhaps the most important one from my PhD studies – that software engineering is a complex interplay between human and technical aspects, which is not easily captured in basic quantifiers, and that software engineering research has to adapt accordingly.

Later, we encapsulated some of the experimentation experiences into a book, where the students vs. professionals issue only was touched upon as threat to conclusion validity, labelled Random heterogeneity of subjects: *For example an experiment with undergraduate students reduces the heterogeneity, since they have more similar knowledge and background, but also reduces the external validity of the experiment, since the subjects are not selected from a general enough population.* (Wohlin et al. 2000, p. 68) Clearly, the topic deserves more nuanced treatment.

Others have worked extensively on the topic, including Falessi et al. (2018a), who confirm that a binary classification of student vs. professional is inappropriate, and as a consequence propose the R3 (Real, Relevant, Recent) characterization scheme to capture experience of subjects. This may be useful to characterize subjects, but the combinatorial explosion of the factors very soon makes experimental designs infeasible. For example, Clarke and O'Connor proposed a reference framework for the whole software development process, consisting of 44 factors and 170 sub-factors (Clarke and O'Connor 2012), of which Experience is one sub-factor of the Personnel factor. Dybå et al. criticized the complexity of their proposal (Dybå et al. 2012): *Assuming only two legal values for each sub-factor this results in a minimum of: 2^170 = 1.5*10^51 combinations of context factors. As a comparison, there are: 1.33*10^50 atoms in the world, which shows the absurdity of the variable oriented logic.*

There seem to be an agreement on that real software engineering contexts are complex. The debate is about what we can do about it.

2 Realism in Software Engineering Experiments

In their paper on research strategies, Stol and Fitzgerald (2015) revisited a classification model by Runkel and McGrath from the social sciences, defined in the 1970's (Runkel and McGrath 1972), which distinguishes between three types of experimental strategies:

- Laboratory experiments,
- Experimental simulations, and
- Field experiments.

In laboratory experiments, they claim, *the investigator deliberately creates a behavior setting not to mirror some naturally occurring behavioral system but rather to highlight selected behavioral processes and certain conditions related to those processes* (Runkel and McGrath 1972, p. 104). In experimental simulations *the activities (behavior) that take place are natural, but the setting in which these occur has been created for the purposes of the study only* (Stol and Fitzgerald 2015). Field experiments, according to Runkel and McGrath, *relies mainly on systematic observation within naturally occurring behavior systems, but also intrudes to the extent of making a deliberate manipulation of one or more variables* (Runkel and McGrath 19729, p. 94). The latter type is similar to case studies and action research in software engineering, and thus we leave it out from the discussion here.

For *laboratory experiments*, the aim is *not* to mimic a real behavioral system, but to focus on generic behavioral processes. The lack of realism is a deliberate design choice to gain better control of the study setting. *Laboratory experiments put maximum emphasis on internal validity, necessarily at some cost in external validity.* (Runkel and McGrath 1972, p. 104) The student vs. professional issue in this type of experimental studies is a matter of whether they resemble the behavioral *process* under study, not that they mimic the behavioral *system*. Behavioral processes are *not* at the same level of abstraction as software engineering processes (e.g. exploratory testing), but rather small components of them (behavioral patterns in exploratory testing); see example by Micallef et al. (2016).

The *experimental simulations* are created to be *like some class of naturally occurring behavior system* (Runkel and McGrath 1972, p. 97). While the laboratory experiment often is defined in a stimuli-response pattern, experimental simulations are based on the dynamics of the system, forcing the subjects to react and act. This implies that the realism of the experimental simulations depends on two major factors (Runkel and McGrath 1972, p. 100): 1) the extent to which the simulated (socio-technical) system resembles the dynamics of the real system, and 2) the extent to which the subjects' relation to the simulated system mirrors the relation between a real actor and a real system. Therefore, the student vs. professional issue is only one aspect of realism. Other important aspects are the realism of the study setting (software engineering artifacts, tools, workspace etc.) and the incentives for the subjects to behave as in a real situation. Runkel and McGrath illustrate the latter with flight simulators vs. real flights, where the 'pilot' has significantly different incentives to keep the aircraft flying in the simulated and real cases, respectively, and thus acts accordingly (Runkel and McGrath 1972, p. 101). Examples of incentives in software engineering experimentation are discussed by Höst et al. (2005) and recently by Dieste et al. (2017a).

In summary, for *laboratory experiments*, the realism is of less importance, since they deliberately are made less real to observe core behavioral processes. For *experimental simulations* the realism is based on several factors, of which the subjects is only one. Thus, the student vs. professional issue is an important factor, but not necessarily on par with the impact it currently has on the assessment of research quality, nor to the contribution of research insights.

3 Implications for Software Engineering Experimentation

I argue that the bottleneck for software engineering research is not whether we use students or professionals as subjects in laboratory experiments. It is rather about how we learn to study software engineering phenomena in their real context. Runkel and McGrath confirm that this is a general trade-off in social sciences: *For each kind of*

problem, no doubt there is an optimum balance between moving the world into the laboratory and moving the laboratory into the world. (Runkel and McGrath 1972, p. 105–106).

To study the complex socio-technical system of software engineering, we cannot resemble the real world in the lab, but have to move more into the real world – that is why my second research method book is on case studies (Runeson et al. 2012). This move naturally turns the research focus more into qualitatively understanding phenomena, over quantitatively demonstrating effects of artificial treatments. The type of research, which compares Method A and Method B with respect to Measure C, is not very realistic in itself. Hence, whether it is conducted with student or professional subjects does not make it realistic either.

Still, software engineering experiments definitely play a role in providing insights – as they did during my PhD studies – but they should be used for research questions which fit the experimental methods, not for their realism. Further, Runkel and McGrath also note that taking knowledge from the lab to the field may imply making the field more lab like. They provide an example from chicken farming. *The spectacular results of the scientific study of the growth of chickens has not been ... the growth of chickens running a free range. Rather, it has been the development of highly controlled environments for raising chickens.* (Runkel and McGrath 1972, p. 105) An effect of laboratory experiments may be that the industry becomes more lab like, although we for other reasons might not want a software engineering office look like a student lab – or a chicken farm!

Finally, *we cannot emphasize more strongly our belief that none of these strategies has any natural or scientific claim to greater respect from researchers than any other.* (Runkel and McGrath 1972, p. 89) Each strategy should be used based on the nature of the problem studied, the current knowledge of the problem and the amount of resources available and be judged accordingly.

References

Clarke P, O'Connor RV (2012) The situational factors that affect the software development process: Towards a comprehensive reference framework. Inf Softw Technol 54(5):433–447. https://doi.org/10.1016/j.infsof.2011.12.003

Dieste O, Fonseca CE, Raura G, Rodriguez P (2017a) Professionals are not superman: Failures beyond motivation in software experiments. In: 5th IEEE/ACM International Workshop on Conducting Empirical Studies in Industry, CESI@ICSE 2017, Buenos Aires, Argentina, May 23, 2017. IEEE, pp. 27–32. https://doi.org/10.1109/CESI.2017.8

Dybå T, Sjøberg DI, Cruzes DS (2012) What works for whom, where, when, and why?: On the role of context in empirical software engineering. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '12. ACM, New York. pp. 19–28. https://doi.org/10.1145/2372251.2372256

Falessi D, Juristo N, Wohlin C, Turhan B, Münch J, Jedlitschka A, Oivo M (2018a) Empirical software engineering experts on the use of students and professionals in experiments. Empir Softw E. https://doi.org/10.1007/s10664-017-9523-3

Höst M, Wohlin C, Thelin T (2005) Experimental context classification: incentives and experience of subjects. In: 27th International Conference on Software Engineering (ICSE 2005), 15–21 May 2005, St. Louis, Missouri, USA. pp. 470–478. https://doi.org/10.1145/1062455.1062539

Micallef M, Porter C, Borg A (2016) Do exploratory testers need formal training? an investigation using HCI techniques. In: Ninth IEEE International Conference on Software Testing, Verification and Validation Workshops, ICST Workshops 2016, Chicago, IL, USA, April 11-15, 2016. IEEE Computer Society, pp. 305–314. https://doi.org/10.1109/ICSTW.2016.31

Runeson P, Wohlin C (1998) An experimental evaluation of an experience-based capture- recapture method in software code inspections. Empir Softw Eng 3(4):381–406. https://doi.org/10.1023/A:1009728205264

Runeson P, Höst M, Rainer A, Regnell B (2012) Case Study Research in Software Engineering – Guidelines and Examples. Wiley

Runkel PJ, McGrath JE (1972) Research on Human Behavior – A Systematic Guide to Method. Holt, Rinehart and Winston, Inc

Stol K, Fitzgerald B (2015) A holistic overview of software engineering research strategies. In: 3rd IEEE/ACM International Workshop on Conducting Empirical Studies in Industry, CESI 2015, Florence, Italy, May 18, 2015. IEEE Computer Society, pp. 47–54. https://doi.org/10.1109/CESI.2015.15

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in Software Engineering: An Introduction. The Kluwer International Series In Software Engineering. Kluwer

Inferencing into the Void: Problems with Implicit Populations

Martin Shepperd

1 Introduction

I welcome the contribution from Falessi et al. (2018a) hereafter referred to as F++, and the ensuing debate. Experimentation is an important tool within empirical software engineering, so how we select participants[1] is clearly a relevant question. Moreover, as F++ point out, the question is considerably more nuanced than the simple dichotomy it might appear to be at first sight.

The majority of software engineering experiments to date use students as participants. The 2005 systematic review of Sjøberg et al. (2005) found that less than 10% of experiments used professionals. More recently da Silva et al. (2012) found in a mapping study of replication studies that 59% used students / researchers, 15% used a mix and 12% used solely professionals as participants. Ko et al. (2015) report a decline from a peak of 44% of tool evaluations using at least one professional participant in 2002 to 26% in 2011. They also reported that an astonishing 23% of studies failed to report any information concerning the nature of their participants. So it seems clear that researchers still predominantly use student participants.

Our discipline is entitled software *engineering*. Many of our concerns and challenges relate to scale. Additionally, engineers must work in complex, dynamic and imperfectly understood environments. How representative are students (i.e., how strong is the external validity) and does this matter? Is their use a valid design decision, a pragmatic compromise, or positively misleading?

The remainder of this commentary is structured as follows. In Section 2 I briefly summarize the arguments of F++ and comment on their approach. Next, in Section 3, I take a step back to consider the nature of representativeness in inferential arguments and the need for careful definition. Then I give three examples of using different types of participant to consider impact. I conclude by arguing — largely in agreement with F++ — that the question of whether student participants are representative or not depends on the target population. However, we need to give careful consideration to defining that population and, in particular, not to overlook the representativeness of *tasks* and *environment*. This is facilitated by explicit description of the target populations.

2 Summary of F++

The objective of F++ is principally to a obtain deeper understanding of the representativeness of software engineering experiments conducted using student participants. This is key to consideration of generalisability. In addressing this challenge they make two observations. First, that there may be circumstances when students behave similarly to professionals. Second, students and professionals are not two dichotomous classes, since a student may have prior professional experience, might be working part-time, might be working as an intern, or his or her educational experience might be highly relevant or realistic to phenomena under

[1] Note that the focus is on human-centric experiments so computational experiments (e.g., comparing different machine learners to predict software defects) are excluded.

investigation. Conversely a professional might have limited experience, indeed could theoretically be in transition (i.e., yesterday a student, today an employee). It is difficult to disagree.

They then obtained further evidence via a large focus group of 65 'experts' and followed it up with a survey. However, one cannot help being concerned that not all conclusions strictly follow. For example, the fact that one author has had poor experiences using professionals in experiments with low numbers and high dropout rates and difficulties getting them to adhere to certain techniques does not mean this must be so. By contrast, I, Carolyn Mair and Magne Jørgensen have recently completed a series of experiments with more than 400 professional participants (Shepperd et al. 2018). Clearly the challenges vary, but equally clearly it's not impossible to conduct experiments with large sample sizes. Additionally, if unwillingness to adhere to a particular technique or intervention differs between students and professionals, this would seem to weaken the value of students as proxies.

The proposal to improve the description of subjects beyond merely whether they are professionals or not is useful. However, limiting it to a 3-class system of real, relevant and recent seems generally overly restrictive. Again, context is all, but minimally one might try to use experience as a continuous variable along with other factors such as role, seniority, and so forth.

A parenthetic concern arises from the stated motivation that "we have observed too many times that our papers were rejected because we used students as subjects" (Falessi et al. 2018a). I fear this betrays a driving force behind some aspects of F++'s paper, despite the difficulty that it's unknowable whether their (or anybody else's) papers have been incorrectly rejected.

Moving on, as F++ remark, "every subject sample is representative of a certain population". For meaningful discussion, however, it's important to define the population. Without this, discussion of sampling and representativeness is indeterminate. Consequently, the remainder of this commentary reflects on the ideas of sampling, representativeness and the population into which the researchers are inferring.

3 First Principles

So the question is what do we mean by representativeness and how might this concept be operationalized? Wieringa and Daneva (2015) refer to software engineering experimentation as a sample-based lab-to-field strategy of generalization where research should be aimed at improving the accuracy of claims regarding scope (i.e., the set of phenomena to which the claims apply given the presently-available arguments and evidence).

The reasoning process is one of inference since we wish the scope of ideas or theories to apply more widely than just those that have been experimentally observed. That is we use inference to generalize. In the experimental paradigm we are most familiar with statistical inference. The sample data are used to generalize statistically to a *well-defined* population, called the study population where the sample is a subset of the study population. The sample should be probabilistic, i.e., we know the likelihood of any member being selected. Often we might wish to extend the generalization beyond the study population to a "theoretical population", of which the study population is a subset (Wieringa and Daneva 2015). The theoretical population may be less well-defined than the study population, in terms of a sampling frame so it might be possible to have a list of all software engineers at Company X, but unlikely to have one for all software engineers globally.

Wilkinson and the Task Force on Statistical Inference (Wilkinson 1999) point out that sometimes "the case for the representativeness of a convenience sample can be strengthened by explicit

comparison of sample characteristics with those of a defined population across a wide range of variables". All this requires not only a thorough description of contextual factors but also explicit consideration of the population, or in Wieringa and Daneva's terminology (2015) the scope. If we don't know into what population we're inferencing then it's hard to see how the process can be open to meaningful scrutiny let alone be considered rigorous. Unfortunately, it does not seem to be common practice in empirical software engineering to formally describe the population even if the inferential statistics are described in considerable detail.

This informality also leads to an under-appreciation of the fact that in software engineering the population is not restricted to the human participants (be they professional or otherwise) but also tasks in particular settings. In other words, most experiments are not only concerned with applying different treatments to different participants, e.g., some might use test-driven development and others traditional development, but this also needs to be applied to particular artifacts in a particular environment. When we infer from the experimental results, we want to generalize to more than just one software artifact and in more than one setting.

This implicit view of a population leads to two further difficulties. First, it is hard to judge the quality of the sample. And second, most software engineering samples are are far from probabilistic — be they students or professionals — which considerably undermines any statistical generalization, some form of analogical inferencing remains possible (Ritchie et al. 2013; Wieringa and Daneva 2015) which could augment the generalization arguments.

4 Examples

An interesting example, drawn from experimental economics (Levitt et al. 2010), used real-world experts in a laboratory setting where they were asked to engage in 2-player games to investigate the ubiquity of minimax strategies. However, the authors found different behaviour in the lab from the participants' professional settings (e.g. as professional poker players) and therefore concluded that it's not just the participant but the context and task as well. It is easy to envisage similar situations in software engineering.

A second example, is based on group estimation and Boehm's delphi estimation process. In this experiment (Passing and Shepperd 2003), we used Master's students as proxies for professionals. The estimation task we gave them was relatively trivial and the context was a laboratory rather than the complexities and uncertainties deriving from a large organizational setting. In terms of representativeness or external validity there were considerable weaknesses and this substantially undermined our ability to generalize. The experimental design decisions were solely for practical reasons. As a pilot study the work may have had some value. As means of saying much about interesting software engineering settings it was decidedly limited.

A third example, was an experiment by Salman et al. (2015) [and cited by F++] to explicitly compare behaviors of professionals and students when using test-driven development methods. Interestingly, they conducted the experiment with students in an academic setting and with professionals in a commercial setting. They found both groups performed relatively similarly when in an initial learning situation, but there were other differences such as professionals producing larger but less complex solutions. This suggests that professionals and students need not always differ, nor need they always behave in similar ways. And we won't know unless at least some experiments or studies use professionals. Furthermore, it is unclear to what extent the differences are due to participants, tasks or the setting, or an interaction between all three.

The point from all three examples is that we should consider the representativeness of participants, tasks *and* setting. Equally important we need to be aware of how they *interact*.

5 Summary

Our concern has been experiments, but we really need a broader set of research methods, e.g., simulation, case studies and action research in order to address realism and the practicality that an engineering discipline demands. Also, there are alternatives to statistical inference and in particular generalization by analogy (Wieringa and Daneva 2015). These should be further explored given our almost invariable use of non-statistical sampling.

The discussion thus far has focused on representativeness, but there is also the point that if we want our research to influence practice it is likely to be more influential when we do not use proxies, however representative. In other words, there is also a 'marketing' or communication to practitioners angle. Use of at least some professional participants undertaking realistic tasks in realistic settings is therefore important if we want our research to have impact.

So to conclude, demanding perfection in the design and execution of our experiments is a counsel of despair; there is clear value in using students especially for preliminary experiments. But if the target population is some class of professional, it is hard to see why, if practically possible, it would be disadvantageous to use professional participants. The likelihood that this might entail more effort should not be a reason for not undertaking such empirical work. But underpinning all other issues, more consideration needs to be given to how we sample tasks and environments as well, and how these aspects of our experiments reflect professional 'reality'. These must be addressed explicitly. Otherwise we are in danger of inferencing into an intellectual void.

Acknowledgements I would like to thank the editors of the Empirical Software Engineering journal, Robert Feldt and Tom Zimmerman for the opportunity to publicly contribute to the debate on the choice of participants in experimentation.

References

Falessi D, Juristo N, Wohlin C, Turhan B, Münch J, Jedlitschka A, Oivo M (2018a) Empirical software engineering experts on the use of students and professionals in experiments. Empir Softw E. https://doi.org/10.1007/s10664-017-9523-3

Ko A, Latoza T, Burnett M (2015) A practical guide to controlled experiments of software engineering tools with human participants. Empir Softw Eng 20(1):110–141

Levitt S, List J, Reiley D (2010) What happens in the field stays in the field: Exploring whether professionals play minimax in laboratory experiments. Econometrica 78(4):1413–1434

Passing U, Shepperd M (2003) An experiment on software project size and effort estimation. In: ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2003). IEEE Computer Society

Ritchie J, Lewis J, Elam R (2013) Designing and selecting samples. Sage, Thousand Oaks, CA

Salman I, Misirli-Tosun A, Juristo N (2015) Are students representatives of professionals in software engineering experiments? In: Proceedings of the 37th IEEE International Conference on Software Engineering. IEEE Computer Society, pp. 666–676

Shepperd M, Mair C, Jørgensen M (2018) An experimental evaluation of a de-biasing intervention for professional software developers. In: Proceedings of the 33rd ACM/SIGAPP Symposium On Applied Computing

da Silva F, Suassuna M, Frana A, Grubb A, Gouveia T, Monteiro C, dos Santos I (2012) Replication of empirical studies in software engineering research: a systematic mapping study. Empir Softw Eng 19(3):501–557

Sjøberg DIK, Hannay JE, Hansen O, Kampenes VB, Karahasanovic A, Liborg N-K, Rekdal AC (2005) A survey of controlled experiments in software engineering. IEEE Trans Softw Eng 31(9):733–753

Wieringa R, Daneva M (2015) Six strategies for generalizing software engineering theories. Sci Comput Program 101:136–152

Wilkinson L, the Task Force on Statistical Inference (1999) Statistical methods in psychology journals: Guidelines and explanations. Am Psychol 54(8):594–604

Responses to Comments on "Empirical Software Engineering Experts on the Use of Students and Professionals in Experiments"

Davide Falessi, Natalia Juristo, Burak Turhan, Jürgen Münch, Andreas Jedlitschka, Markku Oivo

1 Introduction

First, it should be noted that we perceive this EMSE commentary initiative in a very positive way. We believe that since research is iterative by nature, no research result can claim or aim to be definitive. As Runeson commented, "the topic deserves more nuanced treatment". Thus, we do see the presence of three commentary papers as a positive impact of the original paper, considering that the last commentary paper at EMSE was 18 years ago

The three commentary papers are full of great ideas. We are taking this opportunity to discuss some of them. Our own commentary aims to be explicative rather than exhaustive because an exhaustive point-by-point answer to all three commentaries would take more space than the regular size of a commentary paper

To avoid misinterpretations, in the remainder of this commentary, we will use single quotes 'like these' to refer to the content of the original paper, and we will use double quotes "like these" to refer to commentary content.

2 Remarks

The comments of Shepperd, as well as those of Sjøberg and Bergersen, question our statement that "we have observed too many times that our papers were rejected because we used students as subjects." We acknowledge the original statement was ambiguous as it blurred the main point that we expected to be understood from the context. We meant that too often reviewers criticize an experiment conducted with students without elaborating further. Our paper aims to raise awareness that the issue on representativeness, as Shepperd and Runeson discussed in their respective comments, is much more complex than just a simple dichotomy (students = bad; professionals = good). Critics on the type of experimental subjects should consider the goal of an experiment, the type of students, the type of professionals, the type of technology, the targeted population, and so on; see our original paper for further elements. In summary, students should not be treated as second-class experiments participants by reviewers when evaluating an experiment; instead, their use should be assessed case by case.

The commentaries of both Shepperd, and Sjøberg and Bergersen, refer to an overall superiority of experiments with professionals over those with students, despite that they both confirm this is not true for all cases. Our general perspective is in line with that of Runeson: "the aim of an experiment is not to mimic a real behavioral system, but to focus on generic behavioral processes. The lack of realism is a deliberate design choice to gain better control of the study setting." His main argument, which we agree upon, is that "for laboratory experiments, the realism is of less importance, since they deliberately are made less real to observe core behavioral processes." This is in contrast to Shepperd's comment that "if the target population is some class of professionals, it is hard to see why, if practically possible, it would be disadvantageous to use professional participants." Our paper does not claim that experiments with professionals are useless and we should forget about them. Experiments with professionals do not need to be defended because their benefits are widely accepted in the

community. Our paper claims that experiments with students also are necessary to study software development in the laboratory. Our other goal is to remind that professionals also have drawbacks as experimental subjects. Section 4.2 of the original paper reports five different drawbacks of using professionals rather than students. Our main goal was to make readers aware that there might be countless reasons why experiments that used professionals as experimental subjects could have less validity than those that used students. One of our paper's main messages was that deciding on the validity of the participants in an experiment needs to be done through a case by case evaluation, rather than just the golden rule of "professionals always superior to students." While Shepperd does not criticize any of the specific drawbacks we discussed, Sjøberg and Bergersen do so. Thus, below, we clarify and answer to specific critiques provided by Sjøberg and Bergersen.

In Section 1, Sjøberg and Bergersen report on "hypothesized […] curves of the undergraduate and graduate student populations" of "density of skills." These hypothesized curves are used as main arguments against our conclusions. First, there are several threats to validity in their undefined definition of skills and their undefined density measure. Second, it is hard to know in advance which skills benefit which technology. For example in two papers, we expected certain skills to benefit the TDD technology but the results say otherwise (Dieste et al. 2017; Fucci et al. 2015). We do believe that sound empirical data is stronger than "hypothesized" data. In fact, we note that in the original paper, we report the results of a rigorous focus group study rather than the pure opinions of the authors or hypothesized data. We recognize that our empirical procedure is not perfect; for example, few experts explicitly rejected our survey invitation. Our main point here is that it is irrational to challenge empirical results with hypothesized results.

In Section 2, Sjøberg and Bergersen argue against our claim that professionals 'are not as committed/responsible in an experiment as in their daily work.' Sjøberg and Bergersen claim that both professionals and students have deadlines and hence they are both able to meet experimental deadlines. We based such statement on our own experience conducting experiments with professionals and with students. It seems that our experience was different from that of Sjøberg and Bergersen. However, that does not mean they are right, and we are wrong, or vice versa. Both perspectives might be part of the same reality. In other words, sometimes professionals behave better than students in certain experiments, and sometimes the opposite happens. Such veering reality endorses one of the main messages of our paper: much more research is needed on this topic since the reality is not at all as simple as the naive belief on the superiority of professionals over students for every experiment.

Regarding the proposed R3 schema, we did not aim to provide a definitive solution, just a valid starting point to make reader aware that "relevant subject experience" for an experiment is not an easy construct to measure, and hence more research is needed. The fact that both Shepperd, and Sjøberg and Bergersen, judge it as too simple while Runeson as too complex speaks for the difficulty of the issue and suggests that the proposed R3 can be a good practical compromise.

Last but not least, Sjøberg and Bergersen base their entire commentary on a strict difference between students and professionals, as if it would be possible to draw a clear line separating the two groups. Both Runeson and Shepperd agree with one of the main messages of our paper: boundaries are fuzzy. The main conclusion of the original paper is that 'The key is to understand which developer population portion is being represented by the participants in an experiment.' For example, professionals working on low-level code might not be representative of professionals working on software architecture, even in the same company.

3 Conclusion

In conclusion, we recommend taking extreme care while judging any type of validity based on whether subjects have been recruited while at the university or working for a company.

In providing some examples in which professionals could be worse experimental subjects than students, we aimed to raise awareness that too often we use the autopilot to decide on the validity of experiments based on subject types. We wanted to call for more empirical research on the representativeness of students and professionals, on the characterization of experimental subjects, and on the various attributes (motivation, treatment adherence, etc.) that can play a role on how a subject behaves in an experiment. We believe our community has accepted without discussion or evidence to simply divide subjects into two categories: students and professionals. Gladly we have already reached some of our goals, since our paper has apparently raised awareness, resulting in these commentaries from the experts.

Acknowledgement We would like to thank Robert Feldt and Thomas Zimmermann for having provided us with the possibility to reply to the critiques.

References

Dieste O, Aranda AM, Uyaguari F, Turhan B, Tosun A, Fucci D, Oivo M, Juristo N (2017) Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study. Empir Softw E.

Fucci D, Turhan B, Juristo N, Dieste O, Tosun-Misirli A, Oivo M (2015) Towards an operationalization of test-driven development skills: an industrial empirical study. Inf Softw Technol 68(12):82–97.