What Do Software Engineers Care About? Gaps between Research and Practice

Vladimir Ivanov
Innopolis University, Russia
v.ivanov@innopolis.ru

Alan Rogers
Innopolis University, Russia
a.rogers@innopolis.ru

Giancarlo Succi
Innopolis University, Russia
g.succi@innopolis.ru

Jooyong Yi
Innopolis University, Russia
j.yi@innopolis.ru

Vasilii Zorin
Innopolis University, Russia
v.zorin@innopolis.ru

ABSTRACT

It is a cliché to say that there is a gap between research and practice. As the interest and importance in the practical impact of research has been growing, the gap between research and practice is expected to be narrowing. However, our study reveals that there still seems to be a wide gap. We survey software engineers about what they care about when developing software. We then compare our survey results with the research topics of the papers published in ICSE/FSE recently. We found the following discrepancy: while software engineers care more about software development productivity than the quality of software, papers on research areas closely related to software productivity—such as software development process management and software development techniques—are significantly less published than papers on software verification and validation that account for more than half of publications. We also found that software engineers are in great need for techniques for accurate effort estimation, and they are not necessarily knowledgable about techniques they can use to meet their needs.

CCS CONCEPTS

•Social and professional topics →Industry statistics; •Software and its engineering →Software creation and management;

KEYWORDS

Software Engineering Research and Practice, Survey

1 INTRODUCTION

Industrial impact has been gaining attention of many researchers nowadays. Industry tracks, such as ESEC/FSE industry track, have provided for researchers pathways to industrial impact. Typically, researchers team up with industry partners and apply research results to real-world problems of industry. As a result, the doorway to tech-transfer has widened – research results developed in academia are being transferred to industry through this doorway.

ESEC/FSE'17, Paderborn, Germany

DOI: https://doi.org/10.1145/3106237.3117778

In this paper we consider the opposite direction, that is, the transfer of industry problems to academia. In other words, we analyze whether software engineering researchers are solving problems that industry practitioners—more specifically, software engineers–care about, and, especially, the most urgent. Given limited resources available for research as compared to the gigantic size of software industry, researchers need to be strategic in order to maximize their impact. We are *not* arguing here that researchers should tackle only the problems industry practitioners care about. The traditional research effort of constantly exploring and developing new ideas without being necessarily constrained by industry needs should continue to be made. However, if the aim of a researcher is to make an impact on industry, understanding what practitioners care about can be a useful guideline to achieve the aim.

Pushing Research to Practice. The ACM SIGSOFT Impact project assesses the impact of software engineering research on software engineering practice [24]. Under this umbrella project, diverse software engineering research fields—such as programming languages [28], software configuration management [10], runtime assertion checking [7], middleware technology [9], and code inspections and reviews [27]—have been assessed about their impact on practice. Inspired by the Impact project, Lo et al. [19] performed a lightweight survey with software engineers working in Microsoft to investigate how software engineers perceive software engineering research. The assessment of these previous studies is generally positive: software engineering research has made an impact on software engineering practice [24], and the current software engineering research seems generally relevant to software engineering practices exercised in Microsoft [19].

Pulling Practical Needs from Industry. While these previous studies investigate the flow from research to practice as to how impactful and healthy the research-to-practice flow is, the opposite direction of the flow—from practice to research—is relatively rarely studied. In other words, there has been little research done on the issue of "what industry wants from research"—incidentally, however, ICSE 2011 hosted a panel with this same title. The following excerpt from the abstract of the panel casts a different shade from the aforementioned studies: *"Half of the people who attended the first ICSE in 1975 came from industry, but by 2010, industry participation was less than 20%. This lack of participation hurts both sides."*

To remedy this situation, not only should research results be "pushed" into the industry, but also it is important to "pull" the needs of industry. To investigate what industry needs–in particular,

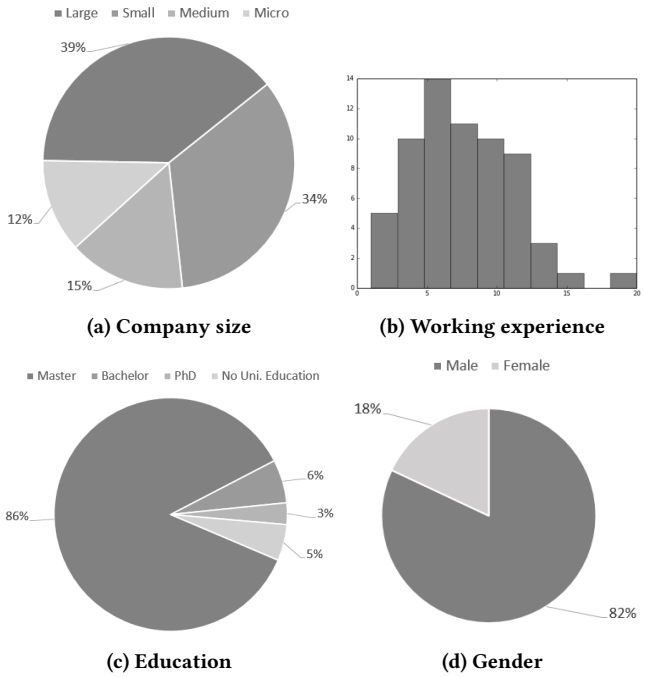
what software engineers need, we conduct a face-to-face survey with 67 software engineers from diverse companies. Then, we compare the needs of software engineers in the field revealed through our survey with the current research landscape we extract by a literature review. To our knowledge, this is the first work in the software engineering community that compares what software engineers care about and what research topics have been presented in recent prominent software engineering conferences.

Our Contributions. Our contributions are as follows:

- *Survey Subjects:* The subjects of our investigation consist of software engineers from multiple companies located in Innopolis, a high-tech city in Russia. While there have been several previous empirical studies that survey the employees of large corporations such as Microsoft [17, 19], developers working in numerous smaller companies have been relatively rarely studied. As compared to previous studies, our survey subjects more closely represent numerous software engineers working in the high-tech regions of the world, such as Sophia Antipolis in France and Shenzhen in China.
- *Survey Administration:* Our survey responses were collected through face-to-face survey sessions, not via a simple on-line or mailed-in questionnaire.
- *Our Finding 1:* According to our survey results, software engineers are in great need for appropriate techniques for more accurate effort estimation. More than half of our survey participants answered that inadequate effort estimation is the biggest obstacle for them.
- *Our Finding 2:* Software engineers care more about development productivity than the quality of software, according to our survey. We also found significantly wide gap among research areas of the papers published in recent ICSE and FSE. More alarmingly, papers on research areas closely related to software productivity—such as software development process management and software development techniques—are significantly less published than papers on software verification and validation that account for more than half of publications.
- *Our Finding 3:* We also found that software engineers in the field are not necessarily knowledgable about techniques they can use to address their obstacles. Software engineers who participated in our survey do not seem to be aware of techniques they can use for effort estimation.

2 RELATED WORK

There have been several studies that show the importance of understanding the needs of practitioners, including [6, 8, 14, 29, 30]. In particular, Misirli et al. [22] reported how they selected research topics in their collaboration with six Finnish software organizations. After understanding the needs of practitioners, they selected TDD (Test-Driven Development) as a collaboration topic. Overall, the practitioners who participated in their study showed high interest in development methods such as TDD and pair programming. Garousi et al. [12] proposed guidelines about how to select research topics for industry-academia collaborations, based on their experience in years of collaboration with their industry partners. Their guidelines start with meeting up with industry partners to catch their needs. Meanwhile, in [13], challenges in industry-academia



(a) Company size (b) Working experience

(c) Education (d) Gender

Figure 1: Survey participants demographics: (a) company size, (a) working experience, (c) education and (d) gender

collaborations and best practices to address these challenges are identified through a systematic literature review. The first two challenges in their challenge list is "Results produced through research are not relevant for practice" and "Researchers do not understand the relevant problems from an industry point of view". The best practices to address these challenges include: "Run workshops and seminars (give access to industry relevant problems)" and "Work in (as) a team (collaboration leads to dissemination and transfer)". Advocating the guidelines of earlier studies, we survey the needs of software engineers. Furthermore, we compare their needs with recent research landscape.

3 SURVEY METHOD AND ITS VALIDITY

Designing an effective survey is challenging; the way questions are defined, organized, and laid down may influence significantly the overall result. To address such limits, we followed the rules found in the literature and similar experiences [3–5, 16, 18, 21, 32, 33]. In particular, we have put a significant care in preventing biases in the responses and confirmation bias, based on [11, 25]. We also use redundancy and replication and avoid leading questions. Moreover, to ensure quality responses, the survey was first sent to the participants, then a member of the research team interviewed them face-to-face, recorded the results in a written document, shared the written document back to the participants to ensure that the right information was captured, and corrected the results if required. Each interview lasted about one hour and the overall preparation for it and followup check required about two hours.

As typical in any surveys, there are threats to the external validity; that is, there is a question about how representative our respondents are. To address this threat, and therefore to increase

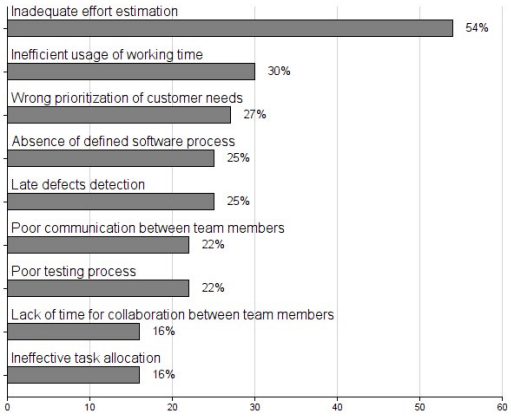


Figure 2: What are the three biggest *obstacles* that affect your ability to deliver software?

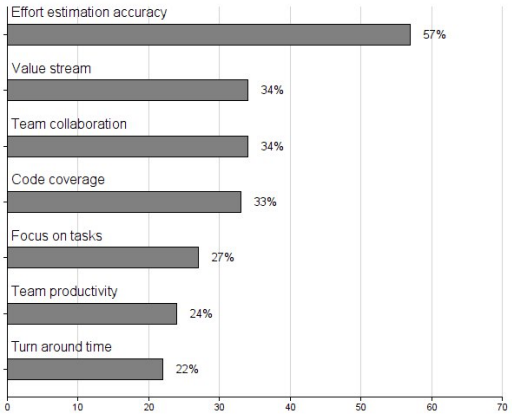


Figure 3: Which three internal improvement aims for your team will you set for the next project?

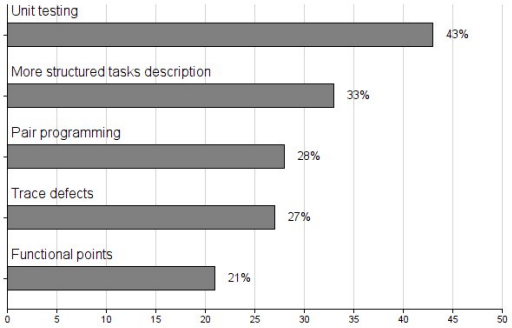


Figure 4: Which three innovations will you introduce to your software process for the next project?

the external validity of our findings, we have followed the best practices found in the literature [15, 20, 31]. In particular, we drew our survey participants from multiple companies in the city of Innopolis, a high-tech city in Russia. The industry there consists of a few large companies ($\geq$ 250 employees) and more number of smaller companies. About half of employees work in large companies, a quarter are small (10 – 49), the rest split between medium size (50 – 249) and micro-sized (< 10). The companies in Innopolis typically employ male software engineers who hold a graduate degree in a STEM field and have more than 5 years of industrial experience.

We contacted 101 software engineers. 67 of them agreed to participate; Figure 1 shows their demographics. Software engineers from at least 44 companies[1] of different sizes participated in our survey—39% from large firms, 15% from medium, 34% from small, and 12% from micro. The working experience ranges from 1 to 20 years, with a median value of 7. Regarding education demographics, 86% had a MS degree, 3% a PhD degree, 6% a BS degree, and 5% no university education. Finally, 82% of the participants were men, 18% women. These demographics tend to reflect the overall structure of the industry in Innopolis.

4 SURVEY RESULTS

4.1 Difficulties Software Engineers Are Experiencing

We asked several question to understand the difficulties software engineers are experiencing. The first question we asked is: "What are the three biggest obstacles that affect your ability to deliver software?" Figure 2 shows the results. More than half of our survey participants said *inadequate estimating efforts*—efforts are also often called work-hours—is one of the major obstacle. Other major obstacles include *inefficient usage of working time*, *wrong prioritization of customer needs*, *absence of defined software process*, and *late defects detection*.

We also asked the following question: "Which three internal improvement aims for your team will you set for the next project?" Notice the subtle difference of this question from the previous question. While with the previous question, we asked the survey participants only to identify obstacles (regardless of whether those obstacles will be address in the future), this second question asks the participants to identify what they want to improve in the next project. This question can help us identify obstacles software engineers have the will to tackle. Figure 3 shows the results. Again, our survey participants showed the keenest interest in improving effort estimation — 57% of the participants answered they want to improve effort estimation. According to our survey results, software engineers strongly feel that there must be improvement in effort estimation.

It is noteworthy that difficulties in controlling software quality seem to be less of a concern to software engineers than difficulties in effort estimation. In Figure 2, only 25% of the participants consider late defects detection as a major obstacle, while inadequate effort estimation is considered a major obstacle by 54% of the participants. Similarly, in Figure 3, more participants want to improve the accuracy of effort estimation than code coverage (57% vs. 33%).

> Software engineers seem to be in great need for improving effort estimation.

4.2 Techniques Software Engineers Want to Use

Figure 4 shows the result for the following question: "Which three innovations will you introduce to your software process for the next project?" This question helps us understand whether the survey

[1] Some participants refused to expose the company they work in.

participants are aware of methods/techniques they can use to tackle the difficulties they are experiencing. Among the five techniques we presented, our participants chose *unit testing* most frequently. Since during our face-to-face survey sessions, many participants said that they could not find options they wanted to choose in the multiple choice options we presented, we asked those participants to suggest additional techniques they want to introduce, and about 60% of our participants (39 out of 67) suggested additional techniques. The two most common answers are (1) techniques related to DevOps and testing such as continuous integration/delivery and (2) code review and the use of standard coding style.

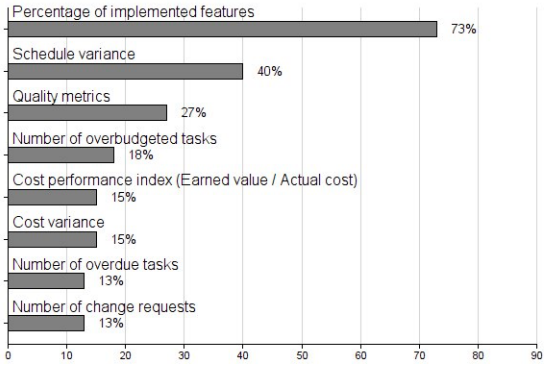
Given that in the previous questions, our survey participants expressed their desire to improve effort estimation the most, our results for this question appear contradictory; the same survey participants seem more interested in code quality than in effort estimation. We suspect that this may be due that our participants are not aware of effective techniques for effort estimation. Some of our survey informants provided general description about what they can try to improve effort estimation such as, "effort estimation by a development team" and "log time spent to execute tasks", but they could not provide techniques to use. This is understandable given that there are not many well-known techniques for effort estimation, as compared to other areas such as software quality where techniques such DevOps and unit testing are widely known. It is also noteworthy that our survey participants least frequently chose an option, "Use function points to estimate tasks". Function points are well-studied productivity measure developed in IBM [1, 2]. By measuring function points based on user requirements, efforts needed to develop a software product can be estimated. Despite high interest in effort estimation, our survey participants showed little interest in using function points in their projects. However, it is not out claim that function points are not useful, because our survey participants may not be knowledgeable about function points.

> Software engineers in the field are not necessarily knowledgable about techniques they can use to address obstacles they experience. We observed mismatch between the biggest obstacle software engineers have (viz., effort estimation) and the technique they are most interested in (viz., unit testing).

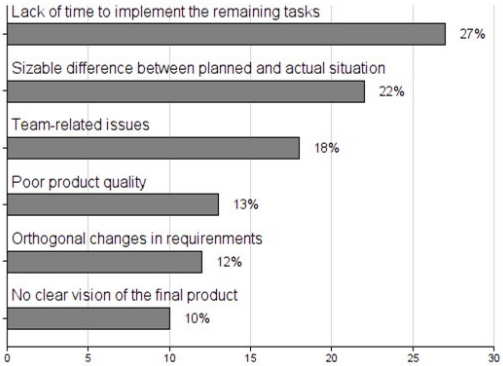
4.3 What Software Engineers Care About

It is effort estimation in which the majority of software engineers who participated in our survey are having difficulties when conducting software development (see Section 4.1). The reason software engineers feel that effort estimation should be improved is likely to be related to how software engineers are currently performing software development process. In this subsection, we show the survey results for questions about what our survey participants care about during software development process for the success of software projects. More specifically, we ask the following two questions related to each other.

(1) What information do you use to ensure that the project will be finished successfully?
(2) What information or events signalize that you will definitely fail the project?



(a) What information do you use to ensure that the project will be finished successfully?



(b) What information or events signalize that you will definitely fail the project?

Figure 5: Responses to the two questions to know what software engineers care about.

Figure 5(a) and 5(b) show the results of these two questions, respectively. In our survey, software engineers care most about *timely implementation of software*. More specifically, 73% of our survey participants pay attention to *percentage of implementation features* to ensure the success of the project, as shown in Figure 5(a). Conversely, failure of a project is conceived most often when when there is not sufficient time to implement the remaining tasks, as shown in Figure 5(b). In comparison, software engineers do not seem to be less concerned about software quality. In Figure 5(a), "quality metrics" is chosen less frequently than "percentage of implementation features". Similarly, in Figure 5(b), "poor product quality" is chosen less frequently than "lack of time to implement remaining tasks".

> Software engineers seem to care more about development productivity (timely implementation of their software products) than the quality of software.

We suspect that our two survey results—first, software engineers are in need for improving effort estimation, and second, software engineers pay great attention to timely implementation for the success of a project—are related to each other. Software engineers who are pressured to keep up with implementation schedule are likely to desire to improve effort estimation, because adequate effort

estimation can help software engineers organize their development activities efficiently, leading to timely implementation of software.

5 THE LATEST RESEARCH LANDSCAPE IN SOFTWARE ENGINEERING

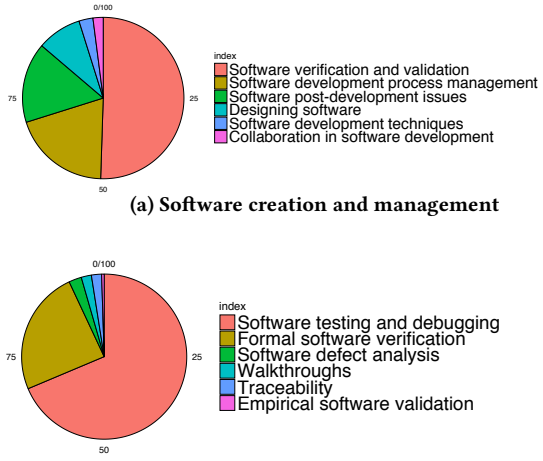
Software engineering research has been rooted in the practical needs of industry since its beginning. At the first (according to [26]) conference on software engineering, in 1968, was written in the section on "Background of Conference" in the conference proceeding [23]:

> *The Conference was to shed further light on the many current problems in software engineering, and also to discuss possible techniques, methods and developments which might lead to their solution. It was hoped that the Conference would be able to identify present necessities, shortcomings and trends and that the findings could serve as a signpost to manufacturers of computers as well as their users.*

Are contemporary software engineering researchers fulfilling the hope of their senior researchers by providing "signpost" for the problems industry practitioners are facing with? While we do not have sufficient data to study this question, our survey results enable studying the following more specific related question: Are software engineering researchers conducting their research on the problems software engineers care about? In other words, *are there any gaps between research efforts and practitioners' needs?*

To answer this question, we conduct the following targeted literature study. We collect the list of the papers published in ICSE (International Conference on Software Engineering) and FSE (Foundations of Software Engineering)—the two prominent conferences on software engineering—for the last three years. We analyze the papers published in the research track of ICSE and FSE, since the goal of our literature review is to draw the recent research landscape. Almost all papers recently published in ICSE and FSE contain CCS (the ACM Computing Classification System) index terms by which the research areas of papers can be classified systematically. CCS is a hierarchical classification system (i.e., a taxonomy), and we analyze the distribution of the following six subcategories under CCS → Software and its engineering → Software creation and management: (1) Designing software, (2) Software development process management, (3) Software development techniques, (4) Software verification and validation, (5) Software post-development issues, and (6) Collaboration in software development, excluding only the "Search-based software engineering" subcategory.

The results of our analysis (Figure 6(a)) evidences that the research results in ICSE and FSE have been highly skewed toward Software verification and validation, more than half of the published papers. Publications in the Software development process management and Software post-development issues account for 20% and 16%, respectively. Meanwhile, publications in Designing software, Software development techniques, and Collaboration in software development only account for 9%, 3%, and 2%, respectively. Given that Software verification and validation accounts for more than half of the publications, we further investigate the distribution of the subcategories of the Software verification and validation category, and Figure 6(b) shows the results. Again, large imbalance in research categories is observed. More than 80% of papers are published in two very closely related fields, viz., Software testing and debugging (69%) and Formal software verification (24%). Meanwhile, research topics that are more closely related to software validation, viz., Walkthroughs and Traceability, account for only about 4%.



(a) Software creation and management

(b) Software verification and validation

Figure 6: The distribution of ICSE/FSE papers in the last three years under (a) the "Software creation and management" category, and (b) its most popular subcategory, "Software verification and validation", respectively. In each pie chart, the legend and slices are sorted by percentage in descending order.

Should this large imbalance in research areas be justified? This is a question worth discussing, we believe. While active research on software verification and validation should be continued given the importance of high quality of software in the modern digital society, we observe a misalignment between the current research landscape and what practitioners care about. According to our survey results, software engineers care more about development productivity than the quality of software. While research on software verification and validation may ultimately contribute to improving the development productivity of software engineers (e.g., advanced debugging techniques can help software engineers fix bugs more quickly than now), it is in the end more directly related to improving software quality. Meanwhile, there are other research topics that are more directly related to software development productivity. For example, in the CCS taxonomy, the Software development process management category covers "Rapid application development" and "Agile software development" that directly aim to improve software development productivity. Similarly, research on Software development techniques can make a significant impact on software development productivity, as was the case of the object-oriented development technique. It is alarming that software engineering researchers nowadays are neglecting the field of Software development techniques: only 3% of publications! We believe research areas other than software verification and validation should deserve further attention from researchers, given that software engineers care not only about software quality, but even more about software development productivity.

6 THREATS TO VALIDITY

Empirical research is subject to threats to the validity of its findings. We have already discussed in Section 3 the measures we have undertaken in designing the survey to limit the threats to its internal validity. Moreover, with reference to external validity, we have presented the data of our respondents, showing that they reflect quite faithfully the industry of Innopolis. The industry structure and demographics of Innopolis seem similar to other highly industrialized parts of the world such as Silicon Valley, Sophia Antipolis in France, and Shenzhen in China; these places—usually filled with a few large companies and a larger number of smaller companies—tend to be male-dominated, and their populations are usually highly educated. Still, we acknowledge that to generalize our results, similar surveys with software engineers in different regions need to be conducted. Similarly, to generalize our analysis results on the current research landscape, further data from other conferences and journals should be considered. Lastly, we would like emphasize that our subjects are sampled from multiple companies of diverse sizes (Figure 1(a)), not from a single corporation such as Microsoft on which many previous studies (e.g., [17, 19]) rely to recruit survey participants.

7 CONCLUSION

In this work, we have exhibited a wide gap between the problems practitioners are facing with and the research topics discussed in the prominent software engineering conferences. We hope that our findings will catalyze discussions about whether the current research landscape is healthy for the software engineering community in the long run. To narrow the gap between research and practice, the software engineering community not only needs the ACM SIGSOFT Impact project [24], but also additional communal efforts to understand the needs of software engineers across the globe. The necessary next steps are (1) to replicate our survey in other regions of the world to obtain a more global picture about what software engineers care about, and (2) to understand the processes driving software engineering research that sometimes seems away from the claimed need of the software industry.

ACKNOWLEDGMENTS

We thank Innopolis University for generously funding this research.

REFERENCES

[1] Allan J. Albrecht. 1979. Measuring application development productivity. In *Proceedings of IBM Applications Develop. Symp.* 14–17.

[2] Allan J. Albrecht and John E. Gaffney Jr. 1983. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Trans. Software Eng.* 9, 6 (1983), 639–648.

[3] Victor R Basili. 1992. Software modeling and measurement: the Goal/Question/-Metric paradigm. (1992).

[4] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 1994. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*. Wiley.

[5] Trevor G Bond and Christine M Fox. 2013. *Applying the Rasch model: Fundamental measurement in the human sciences.* Psychology Press.

[6] Justin Clark, Chris Clarke, Stefano De Panfilis, Giampiero Granatella, Paolo Predonzani, Alberto Sillitti, Giancarlo Succi, and Tullio Vernazza. 2004. Selecting components in large COTS repositories. *Journal of Systems and Software* 73, 2 (2004), 323–331.

[7] Lori A. Clarke and David S. Rosenblum. 2006. A historical perspective on runtime assertion checking in software development. *ACM SIGSOFT Software Engineering Notes* 31, 3 (2006), 25–37.

[8] Enrico di Bella, Ilenia Fronza, Nattakarn Phaphoom, Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. 2013. Pair Programming and Software Defects–A Large, Industrial Case Study. *IEEE Transactions on Software Engineering* 39, 7 (2013), 930–953.

[9] Wolfgang Emmerich, Mikio Aoyama, and Joe Sventek. 2007. The impact of research on the development of middleware technology. *ACM Trans. Softw. Eng. Methodol.* 17, 4 (2007), 19:1–19:48.

[10] Jacky Estublier, David B. Leblang, André van der Hoek, Reidar Conradi, Geoffrey Clemm, Walter F. Tichy, and Darcy Wiborg Weber. 2005. Impact of software engineering research on the practice of software configuration management. *ACM Trans. Softw. Eng. Methodol.* 14, 4 (2005), 383–430.

[11] Adrian Furnham. 1986. Response bias, social desirability and dissimulation. *Personality and individual differences* 7, 3 (1986), 385–400.

[12] Vahid Garousi and Kadir Herkiloglu. 2016. Selecting the Right Topics for Industry-Academia Collaborations in Software Testing: An Experience Report. In *ICST*. 213–222.

[13] Vahid Garousi, Kai Petersen, and Baris Özkan. 2016. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information & Software Technology* 79 (2016), 106–127.

[14] Andrejs Jermakovics, Alberto Sillitti, and Giancarlo Succi. 2011. Mining and visualizing developer networks from version control systems. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, 24–31.

[15] Yasser Khazaal, Mathias van Singer, Anne Chatton, Sophia Achab, Daniele Zullino, Stephane Rothen, Riaz Khan, Joel Billieux, and Gabriel Thorens. 2014. Does Self-Selection Affect Samples'Representativeness in Online Surveys? An Investigation in Online Video Game Research. *Journal of Medical Internet Research* 16, 7 (July 2014), e164.

[16] Jon A Krosnick and Stanley Presser. 2010. Question and questionnaire design. *Handbook of survey research* 2 (2010), 263–314.

[17] Paul Luo Li, Andrew J. Ko, and Jiamin Zhu. 2015. What Makes a Great Software Engineer?. In *ICSE*. 700–710.

[18] Petra Lietz. 2008. *Questionnaire design in attitude and opinion research: Current state of an art.* Citeseer.

[19] David Lo, Nachiappan Nagappan, and Thomas Zimmermann. 2015. How Practitioners Perceive the Relevance of Software Engineering Research. In *ESEC/FSE*. 415–425.

[20] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the Comprehension of Program Comprehension. *ACM Trans. Softw. Eng. Methodol.* 23, 4, Article 31 (Sept. 2014), 37 pages.

[21] Frank Maurer, Giancarlo Succi, Harald Holz, Boris Kötting, Sigrid Goldmann, and Barbara Dellen. 1999. Software process support over the Internet. In *Proceedings of the 21st international conference on Software engineering*. ACM, 642–645.

[22] Ayse Tosun Misirli, Hakan Erdogmus, Natalia Juristo Juzgado, and Oscar Dieste. 2014. Topic selection in industry experiments. In *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry, CESI 2014, Hyderabad, India, June 2, 2014.* 25–30.

[23] Peter Naur and Brian Randell (Eds.). 1969. *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch Germany 7-11 Oct. 1968.*

[24] Leon J. Osterweil, Carlo Ghezzi, Jeff Kramer, and Alexander L. Wolf. 2008. Determining the Impact of Software Engineering Research on Practice. *IEEE Computer* 41, 3 (2008), 39–49.

[25] Philip M. Podsakoff, Scott B. MacKenzie, Jeong Yeon Lee, and Nathan P. Podsakoff. 2003. Common Method Biases in Behavioral Research: A Critical Review of the Literature and Recommended Remedies. *Journal of Applied Psychology* 88, 5 (October 2003), 879–903.

[26] B. Randell. 1979. Software Engineering in 1968. In *Proceedings of the 4th International Conference on Software Engineering (ICSE '79)*. 1–10.

[27] H. Dieter Rombach, Marcus Ciolkowski, D. Ross Jeffery, Oliver Laitenberger, Frank E. McGarry, and Forrest Shull. 2008. Impact of research on practice in the field of inspections, reviews and walkthroughs: learning from successful industrial uses. *ACM SIGSOFT Software Engineering Notes* 33, 6 (2008), 26–35.

[28] Barbara G. Ryder, Mary Lou Soffa, and Margaret M. Burnett. 2005. The impact of software engineering research on modern programming languages. *ACM Trans. Softw. Eng. Methodol.* 14, 4 (2005), 431–477.

[29] Marco Scotto, Alberto Sillitti, Giancarlo Succi, and Tullio Vernazza. 2004. A relational approach to software metrics. In *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 1536–1540.

[30] Giancarlo Succi, Luigi Benedicenti, and Tullio Vernazza. 2001. Analysis of the effects of software reuse on customer satisfaction in an RPG environment. *IEEE Transactions on Software Engineering* 27, 5 (2001), 473–479.

[31] Gergely Szolnoki and Dieter Hoffmann. 2013. Online, face-to-face and telephone surveys – Comparing different sampling methods in wine consumer research. *Wine Economics and Policy* 2, 2 (2013), 57 – 66.

[32] Nancy Thayer-Hart, Jennifer Dykema, K Elver, NC Schaeffer, and J Stevenson. 2010. Survey fundamentals: A guide to designing and implementing surveys. *Office of Quality Improvement* (2010).

[33] David L. Vannette and Jon A. Krosnick. 2014. *Answering Questions: A Comparison of Survey Satisficing and Mindlessness.* John Wiley & Sons, Ltd, 312–327.