Implications of Resurgence in Artificial Intelligence for Research Collaborations in Software Engineering

Dusica Marijan
Simula Research Laboratory
Norway
dusica@simula.no

Weiyi Shang
Concordia University
Canada
shang@encs.concordia.ca

Rakesh Shukla
Creating Innovators
India
rakeshshukla@acm.org

DOI: 10.1145/3356773.3356813
http://doi.acm.org/10.1145/3356773.3356813

ABSTRACT

Challenges of implementing successful research collaborations between industry and academia in software engineering are varied and many. Differing timelines, metrics, expectations, and perceptions of these two communities are some common obstacles, which need be analyzed and discussed, to discover synergies and strengthen collaborations between researchers and practitioners. In this report, we present insights from the 6th International Workshop on Software Engineering Research and Industrial Practice held at the International Conference on Software Engineering 2019. Specifically, one particular topic dominated the discussion - the resurgence of artificial intelligence and machine learning algorithms in software engineering research and industry practice, and its implications for the collaboration between these two communities. We present takeaways from keynote talks on this subject, insights from paper presentations, and findings from the discussion session.

Categories and Subject Descriptors

D.3.2 [**Software Engineering**]

General Terms

Management, Measurement, Economics, Experimentation, Human Factors, Legal Aspects.

Keywords

Technology transfer, innovation, industry-academia collaboration, research collaboration, software engineering.

1. INTRODUCTION

Successful collaborations between software engineering academia and software industry are known to be challenging for a number of reasons. As observed by Chimalakonda et al. [1], researchers can be unaware of real problems and constraints in practice, whereas practitioners may find themselves unable to adopt existing useful research. Researchers often have a view that practitioners are reluctant to share real industry data due to confidentiality and legal concerns. Practitioners often feel that researchers work on dated or futuristic theoretical challenges which are divorced from today's industrial practice. Researchers believe that practitioners are looking for quick fixes instead of using systematic methods. Practitioners have a view that case studies in research do not represent the complexities of real projects and often dismiss results outright when students are used as test subjects in research. Researchers expect good work to take a few years to generate good publications which may affect a specific domain in an incremental manner. Practitioners expect a quick solution which must pay off immediately. Researchers are more interested in proposing new techniques and tools. Practitioners would appreciate systematic evaluation and comparison of existing techniques and tools in real-world settings.

Software Engineering Research and Industrial Practice (SER&IP) workshop series is a dedicated forum to discuss these and related challenges. Specifically this year, a topic that received great interest for both the academic and industry community revolved around the surge of artificial intelligence (AI) and machine learning (ML) in software engineering, and its implications for research collaboration between researchers and practitioners.

The 6th SER&IP workshop was held at the International Conference on Software Engineering (ICSE) 2019. The workshop featured two eminent keynote talks from industry and academia, four paper presentations, and the "discussion" session. The focus of the interaction was on sharing experiences from both successful and unsuccessful collaborations, summarized in the form of ideas and lessons learned on this subject. We report here the findings and discussions from the workshop, with the goal to provide a better understanding of bottlenecks in collaboration, as seen from both perspectives, to discover synergies, enable alignment, and strengthen the collaborations between these two communities.

2. TAKEAWAYS FROM KEYNOTE TALKS

2.1 Lessons Learned from Engineering AI-infused Applications at Microsoft

Andrew Begel [2] summarized best practices for engineering AI-infused applications in the form of lessons learned from Microsoft teams. The keynote focused on several large scale empirical studies [3, 4, 5] discussing how Microsoft software teams develop AI/ML-based applications. In particular, a nine-stage AI workflow process (Model requirement, Data Collection, Data Cleaning, Data Labeling, Feature Engineering, Model Training, Model Evaluation, Model Deployment and Model Monitoring) was defined by practitioners in the development of AI applications or data science tools. The studies illustrate the challenges that are encountered when such a workflow is adapted into a pre-existing, well-evolved, agile-like software engineering process. The challenges include: i) end-to-end pipeline support, ii) data availability, collection, cleaning and management, iii) education and training, iv) model debugging and interpretability, v) model evolution, evaluation and deployment, vi) compliance, and vii) varied perceptions. Some of the challenges are universal while others may be specifically related to experiences and educational background in the team.

In addition, the study found that education and training are negatively correlated with AI experience, while educating others and tool issues are all positively correlated with AI experiences. Such findings show that people with less AI experience find education and training to be important, while people with higher AI experience find educating others to be important. Moreover, people with higher AI experience rely on tools (from both themselves and others) more. The keynote concluded that the lessons that Microsoft has learned can help other organizations in leveraging AI and ML in building their products.

2.2 Challenges for Software Engineering in a Data Science World

Ahmed E. Hassan gave a critical look at challenges arising from the widespread use of ML in software engineering research and practice. The keynote summarizes the advances of using ML in software engineering research in the past decade, in particular, with the creation of the Mining Software Repositories (MSR) conference and its research community. The keynote provides numerous examples that have already had impact in software engineering practice in industry. Moreover, the keynote critically points out that the use of such sophisticated advances from ML in software engineering research is challenging and with certain risks. For example, ML has been used to identify risky code changes before they are merged to the code base. Data analytics is used to automatically analyze the results from performance testing. The advance of ML and data science tools (e.g., R, WEKA, and Jupyter) has lowered the burden of leveraging such techniques in software engineering research. However, with the ease of using such tools, practitioners may treat ML as a blackbox, without considering its inner complexity. By providing simple illustrative examples, the keynote showed that the suboptimal use of ML is risky. For example, without proper feature engineering, the conclusion of simple data analytics can be misleading. In addition, the inclusion of correlated metric may lead to inconsistent interpretation of statistical models. Furthermore, simple tools of ML and data science may hide the internal dependency and complexity.

The keynote points out that the identified risks should motivate us to consider leveraging the software engineering expertise in ML and data science, such as testing, version control and results explanation. The keynote calls for a careful usage of such ML techniques in the software engineering research and practice. The final message from the keynote indicates that software engineering is essential for data science success and maturity.

3. INSIGHTS FROM PAPER PRESENTATIONS

3.1 How Engineers Perceive Difficulties in Engineering of Machine Learning Systems

As machine learning techniques and their applications gain popularity, Ishikawa et al. [3] discussed a need to identify the essential challenges for the software engineering research community pertaining to support the widespread use of AI in industry and society. The authors point out that the development of engineering disciplines and techniques is crucial in this regard. This is important because ML-based systems are essentially different from the traditional systems, for which such disciplines and techniques are available. For example, the behavior of traditional systems is defined by rules, whereas for ML-based systems, the behavior is inductively derived from training data. In this light, the authors analyzed the most pressing challenges of developing ML-based systems as seen by practitioners, such that research efforts can be focused on providing effective solutions addressing these challenges. The analysis was conducted in the form of a survey targeting 278 practitioners having the experience with developing ML-based systems. Findings from the survey indicate that the three most important quality attributes of ML-based systems that make these systems difficult to engineer are: i) understandability and explainability of outputs, ii) long-term maintenance and adaptation to changes, and iii) handling of various inputs. The three most difficult activities related to the engineering of ML-based systems include: i) decision making with customers, ii) testing and quality evaluation, and iii) debugging. The three top sources of difficulties are: i) the lack of oracle, ii) imperfection, and iii) behavior uncertainty for untested data.

The authors summarized that there is a need to provide precise guidance about ML for non-technical groups, which are confused by the various streams of information available on AI, and therefore may have unrealistic expectations of ML-based solutions. Similarly, new disciplines and empirical studies are required to understand the principles of testing and quality assurance of ML-based systems. Finally, the authors advocated that active research is needed in the field of ML Systems Engineering, developing new paradigms in this area.

3.2 Do Practitioners and Researchers in Software Testing Speak the Same Language

Software testing has gained attention from both researchers and practitioners as it consumes a large portion of software development time. However, Santos et al. [4] argue that practitioners and researchers have different views on what aspects of the testing process are more important and should be further researched and developed. Thus, the authors analyzed the difference of interests between academics and researchers in software testing, to identify the gap, and potentially reduce the gap between these two communities. To gather evidence, a combination of a mapping study, a quantitative study, and a focus group discussions was used. The findings point out that the main focus of researchers is test design, test execution, and test planning. Practitioners, on the other hand, are mainly interested in test design and test execution, and especially on the matter of test automation. One notable difference in interests between these two groups is that researchers are more interested in proposing new techniques and tools, while practitioners care about the evaluation, comparison and discussion of existing techniques and tools. This finding suggests one research direction which can potentially have a significant impact on industry practice, which is the understanding and adapting of existing testing tools and strategies for different industry settings.

The authors also revealed that practitioners seldom search for advice in academic papers, especially regarding tools and frameworks, and they prefer to obtain such information from community forums. When they do look for academic papers, it is usually about AI applied in software testing. Finally, the authors advocate that using research methods that practitioners are familiar with, such as surveys and case studies, would be an effective way to increase the interaction between the two communities. This should be complemented with demonstrations of the usefulness of the findings presented in academic papers.

3.3 Can Gamification of Exploratory Industry-Academia Research Bridge the "Gap"

The "Gap" between software engineering researchers and industry practitioners is a well discussed phenomenon. Walenstein et al. [5] addressed this matter with a generative probabilistic modeling and model evolution through gamification. The hypothesis of the approach is that the exact nature of the problem and solution are unknown. The idea then is to use an adversarial game that represents a collaborative exploratory research process aiming to answer the questions of validity of the technical solution. Threats to the validity of the solution are identified by the threat model, and the evaluation of the solution is performed using both synthetic and authentic data. The project starts with a "minimal viable increment", which includes a simple technical solution and an initial generative model. Through adversarial game playing, research partners iteratively develop more refined generative models that make the technical solution fail, and in response, iteratively more capable technical solutions that adapt to the new challenges found by the generative model. The technical solution becomes more capable to handle cases found in authentic data and the generative model becomes more aligned with the specifics of a real life context.

The authors believe that such gamification approach can help resolve several roadblocks in industry-academia collaboration. They include: i) the confidentiality of industrial data, ii) incompleteness of industrial data which leads to the problem of not having enough data for the analysis and insufficient coverage of the conditions of interests, iii) access and form of industrial data requiring continuous collection, and iv) finally alignment of partners' interests. The authors provide ideas for how their approach can be extended with existing related work, including differential privacy, generative adversarial learning, and probabilistic programming.

3.4 What are Useful Criteria for Extracting Microservices in Practice

Microservice architecture is receiving attention in the development of software systems, due to the range of potential benefits including reduced maintenance effort, increased availability, continuous delivery, and scalability. For this reason, there has been a growing interest in both industry and academia for efficiently migrating existing systems to a microservice architecture. This involves the challenge of extracting microservices from a code base. Carvalho et al. [6] suspected there is a discrepancy between the way academics and industry practitioners extract microservices, so they looked into what criteria for extracting microservices practitioners find useful. They performed an online survey with 15 specialists in this area. The survey used the set of seven criteria found in papers written by researchers and practitioners. They include coupling, cohesion, overhead in communication of extracted microservices, potential of reuse, data dependency in database schema, impact on software requirements, and high level criteria from visual models. The survey revealed that academics' solutions do not fully satisfy the needs of practitioners. Specifically, practitioners often need to satisfy at least four dominant criteria, and consider their tradeoffs. Academic solutions, on the other hand, usually support the analysis of a limited set of criteria. Furthermore, many practitioners found existing tools insufficient or irrelevant for their work. For example, cohesion was found to be the most useful extraction criteria, but proper tool support is not available. Finally, authors highlighted the need for researchers to focus their effort on criteria deemed more important from practitioners' perspective, and encouraged both communities to work together on moving this field forward.

4. FINDINGS FROM DISCUSSION SESSION

After paper presentations, a "Discussion session" followed, which focused on the discussion points raised during paper presentations that the audience would like to discuss further. After initial brainstorming, the discussion mainly focused on the topic: How to evaluate the success of software engineering research in practice (for example in industrial settings)? One of the views from the discussion supports the use of subjective measures, such as interviews and surveys with the stakeholders or industrial practitioners. The counterview, however, pointed out that such measures may be flexible, may not be used to compare and prioritize collaboration, and may be prone to bias. A hybrid measurement may be preferred and may be evaluated as a topic of future research.

5. CONCLUSION

SER&IP 2019 brought together software engineering researchers and industry practitioners to discuss the challenges that arise in their mutual collaboration. The overall message of the discussions made in the workshop is that in order to make a stronger impact on the industry practice in software engineering, academic research needs to be more closely aligned with the needs of practitioners. It was suggested that researchers could focus more efforts on analysis and benchmarking of existing techniques and tools instead of developing new ones. It was also pointed out that research methods, such as surveys and case studies, which practitioners are familiar with, could be effective in increase the interaction between the two communities. A concreate way suggested to better align academia and industry is a gamification approach of exploratory academia-industry research. In addition, a guided schema for a proper evaluation on practice-impacting software engineering research is needed. Finally, both of the keynotes heavily focused on the research and practice of using AI/ML techniques in software engineering, implying the need of more impactful software engineering research within the context of AI/ML.

Overall, the perspectives on the challenges and opportunities for the collaboration between software engineering researchers and practitioners discussed during the workshop were found interesting for the members of both communities. Hopefully, the workshop attendees gained a better understanding of the problems faced by both communities in establishing a joint research collaboration, as well as opportunities to strengthen such collaboration.

6. ACKNOWLEDGMENTS

We thank to all the participants of the workshop, the keynote speakers, the authors and paper presenters for their contribution to the workshop. We thank to the members of the SER&IP 2019 program committee for helping prepare a great program, as well as the organizers of the ICSE 2019 conference, and the ICSE 2019 Workshop Chairs.

7. REFERENCES

[1] Chimalakonda, S., Reddy, Y. R., and Shukla, R. Moving Beyond: Insights from 1st International Workshop on Software Engineering Research and Industrial Practices (SER&IPs 2014). *ACM SIGSOFT Software Engineering Notes*, April 2015.

[2] Begel, A. Best Practices for Engineering AI-infused Applications: Lessons Learned from Microsoft Teams, *Proceedings of the Joint 7th Int. Workshop on Conducting Empirical Studies in Industry and 6th Int. Workshop on Software Engineering Research and Industry Practice,* pp. 1-1, May 2019.

[3] Kim, M., Zimmermann, T., DeLine, R. and Begel, A. The Emerging Role of Data Scientists on Software Development Teams. *In Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 96-107, 2016.

[4] Kim, M., Zimmermann, T., DeLine, R. and Begel, A, "Data Scientists in Software Teams: State of the Art and Challenges," in *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1024-1038, 1 Nov. 2018.

[5] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B. and Zimmermann. T. Software Engineering for Machine Learning: A Case Study. *In Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '10)*. IEEE Press, Piscataway, NJ, USA, 291-300, 2019.

[6] Ishikawa, F. and Nobukazu, Y., How Do Engineers Perceive Difficulties in Engineering of Machine Learning Systems?: Questionnaire Survey, *Proceedings of the Joint 7th Int. Workshop on Conducting Empirical Studies in Industry and 6th Int. Workshop on Software Engineering Research and Industry Practice,* pp. 2-9, May 2019.

[7] Santos, R.E.S, Bener, A.B., Baldassarre, M.T., Magalhaes, C.V.C., Correia-Neto, J.S. and Silva, F.Q.B. Mind the Gap: Are Practitioners and Researchers in Software Testing Speaking the Same Language? *Proceedings of the Joint 7th Int. Workshop on Conducting Empirical Studies in Industry and 6th Int. Workshop on Software Engineering Research and Industry Practice,* pp. 10-17, May 2019.

[8] Walenstein, A. and Malton, A. Generative Modelling Games for Exploratory Industry-Academic Research", *Proceedings of the Joint 7th Int. Workshop on Conducting Empirical Studies in Industry and 6th Int. Workshop on Software Engineering Research and Industry Practice,* pp. 18-21 May 2019.

[9] Carvalho, L., Garcia, A., Assuncao, W., Mello, R. and Lima, M.J. Analysis of the Criteria Adopted in Industry to Extract Microservices, *Proceedings of the Joint 7th Int. Workshop on Conducting Empirical Studies in Industry and 6th Int. Workshop on Software Engineering Research and Industry Practice,* pp. 22-29, May 2019.