This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

Successful engagement of practitioners and software engineering researchers:

Evidence from 26 international industry-academia collaborative projects

Vahid Garousi
Wageningen University

David C. Shepherd
ABB Corporate Research

Kadir Herkiloğlu
HAVELSAN A.Ş.

Abstract:

There has been a recent push to increase the practical relevance and impact of software engineering (SE) research. Even though many practitioners and researchers agree that this change is desirable, only some concrete actions have been taken in the community so far. In this paper, we present our experience in a large number of collaborative research projects (26 projects) which have had practical (industrial) impact. These projects have been conducted in several different countries, have focused on different SE topics (e.g., testing, software maintenance, and documentation), and have spanned over different domains (e.g., embedded software, defense and telecom, robotics). We characterize the industrial needs, contributions, and impacts of the projects. Furthermore, via a participant-observation research approach, the authors analyze their diary reflections recorded during the projects and synthesize their experience into a set of seven lessons learned on how to conduct successful industry-academia collaborations. Our hope is that the evidence and experience provided by our example projects would motivate SE practitioners and researchers to engage more on collaborative projects.

Keywords: Industry-academia collaborations; experience report; lessons learned; applied research

1 INTRODUCTION

Assessing the "impact" of Software Engineering (SE) research on practice (industry) has been an important topic for many years. For example, the ACM SIGSOFT Impact project (www.sigsoft.org/impact.html) was an initiative, conducted between 2000-2008, "*whose goal is [was] to determine the impact of software engineering research upon software engineering practice*". Many believe that, in an applied field such as SE, industrial impact is of outmost importance.

To highlight the importance of industry-academia collaborations (IAC), to discuss success stories and how to "bridge the gap", various workshops and panels are also regularly organized in international conferences, such as a recent 2018 panel entitled "*When are software testing research contributions, real contributions?*" (www.es.mdh.se/icst2018/live), held in the IEEE Conference on Software Testing, Validation and Verification (ICST). The panelists, who were a mix of practitioners and researchers, discussed their opinions about this important issue. Understanding "*real [concrete] challenges of industry*" was mentioned as an important issue in expanding the contributions and impact of SE research. Also, a panel called "*What industry wants from research*" was organized in the International Conference on Software Engineering (ICSE) 2011 in which talks from companies such as Toshiba, Google and IBM were presented.

Yet, in spite of these efforts "*There are [still] gaps between SE research and practice*" [1]. According to the opinion of many practitioners and researchers, e.g., [1, 2], the level of joint IACs in SE is unfortunately low, compared to the amount of activities in each of the two communities. As one senior software architect put (goo.gl/aj55uJ): "*…it's been so long since anything actually relevant to what practitioners do has come out of that environment [academic research], or at least the percentage of things that are useful that come out of that environment is so small*". Various reasons have been discussed for the low ratio of IACs, such as each side having different objectives, industrial problems lacking scientific novelty or challenges, and the low applicability and scalability problems of the solutions developed in the academia [2].

It is good to see that SE practitioners and researchers have recently started to conduct increasingly more IAC together and to report more studies in this context, e.g., [3, 4]. Similar to the empirical SE community, the hope is that, by sharing more experience about successful IACs, and even guidelines for effective IAC, SE practitioners and researchers would be encouraged to focus further on industrial impact.

We should note that, in general, research "impact" has two aspects: academic impact and industrial impact. While we focus on industrial impact in this paper, we should note that academic impact is also equally important. In fact, in academia, success of research papers and projects is often assessed by academic impact, which is primarily measured by citations [5]. Thus, when assessing research papers and projects, it is fair to consider both academic impact and industrial impact.

We should furthermore note that "direct" (explicit) IAC is not the only approach to conduct practically-relevant research. A lot of high-quality research has been reported by SE researchers without direct industrial collaborations, e.g., many techniques to analyze open-source software or to mine software repositories. Although most of those approaches have not been explicitly assessed in industrial settings directly in the papers that have presented them, they are often rooted in "realistic" challenges and have been developed based on realistic assumptions. Thus, it is fair to say that they have high

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

potential to be applicable and useful in industrial settings and may have been used already in such contexts. However, it is often hard to track such "indirect" impacts since relatively much fewer papers (compared to all SE papers) take SE techniques and methods, presented in purely academic papers, and apply them in industry. Furthermore, a lot of solid fundamental research has been reported in SE, e.g., works in the area of theory-building, which help us as the community to better understand SE in practice and to consequently conduct more research with industrial impact.

Many experience papers have been published on the topic of IACs in SE, e.g., [3, 4]. A 2016 systematic literature review (SLR) [6] synthesized the challenges and best practices as reported in those many experience papers. This practitioner-focused article aims to increase the awareness of industry-driven SE research and to provide further concrete examples of how IAC projects are initiated, conducted, challenged and how both sides can do their best to make it work.

It should be noted that two of the authors have published a slightly-related previous paper [7], in which they analyzed the extent of observed challenges and applied best practices in software testing. Our focus in this study is different since we highlight lessons learned, via concrete examples and reflections, instead of analyzing challenges. Also, we expand our dataset of projects from 10 to 26 IAC projects, broadening our focus from testing by adding software maintenance, best practices, and productivity. The experience-based lessons learned that we present in this paper would lead to better IACs and will also increase the chances of impactful research from IACs. Another recent paper [8] analyzed the extent of observed challenges and the applied best practices in another set of 47 projects, conducted by other researchers.

Let us start by reviewing our method for synthesis of experience, reported in this paper.

2 Our method for synthesis of experience and lessons learned

The three authors have participated in a large number of IACs based on their positions in academic organizations, corporate research centers and software companies. The contexts in which they have worked in, the organizational structures, funding programs, research goals and strategies, have often supported conducting IACs projects. For example, while the first author was working in Canada, he established a good record in acquiring grants under the "Engage" and Collaborative Research and Development (CRD) programs by the Natural Sciences and Engineering Research Council of Canada (NSERC). The former was, by definition, an instrument to boost IAC. The later "required" grant proposals to have commitment from industry partners. Both such mechanisms led to several successful and impactful IACs [7].

The other more fundamental issue has been the research "mindset", goal, and strategies of the authors. Each author has independently been passionate about applied SE research with industry impact and has pursued impact via active and close IACs. Throughout their careers, they have either have been in contexts which have supported such a research strategy, or they themselves have taken proactive initiatives to "create" such contexts inside their own research groups.

The (research) method that we used for synthesizing and reporting the experience and lessons learnt from our projects is "participant-observation" [9]. This is a method long used in anthropology, and also more recently in SE, in which the researcher is both an observer and a participant in some activity over time. Participant observations have the unique strength of describing complex aspects of cognition, social interaction, and culture over time; and have been used in SE research, e.g., [10, 11]. For example, in 1989, Donald Knuth published the Errors of TeX [10], a diary study documenting and reflecting on the 850 errors in the TeX toolset that he made over a decade of work. Following the participant-observation method and its guidelines [9] enabled us conduct a rigorous and systematic study in this work. We had kept diary reflections and history of emails during our projects. By systematically analyzing those sources, we synthesize in this paper our experience into a set of the most important lessons learned in IAC.

Furthermore, we use the "case survey" method [12] in which we analyze (survey) each of our IAC projects as a "case" and then conduct meta-analysis to synthesize our experience from the set of projects. By retrospective analysis, we also look back and provide narratives on how the projects were conducted. Now, it is time to review the projects from which we have drawn our experiences.

3 List of the IAC projects under study

This experience paper is based on a pool of 26 IAC projects, as listed in Table 1. For each project, we provide the need (the challenge solved), the provided solution, its industrial impact, assessment of success (from both industrial and academic perspectives), and the number of the resulting papers from each project. Due to space constraints, we do not include in this paper the references to those papers, but they can be found in an online supplementary document [13].

As our research approach, we have used "action-research" [12] in all our projects. For the company size column in this table, small (S) corresponds to companies with 1-100 employees. Medium (M) corresponds to 101-500 employees, and large (L) corresponds to 500+ employees. While most of the projects were successful, we have intentionally included in the pool, less successful (challenged) and even prematurely-terminated projects (such as CA6 and TR2), to convey the message that IACs are not always successful and projects could fail due to different types of challenges [6]. Thus, the pool has a good level of diversity in terms of various factors. Projects starting with "CA" and "TR" were conducted in Canada and Turkey, respectively. Those starting with "ABB" were conducted in the context of the ABB, a large multi-national company. While

most projects were independent from one another, there were connections among some projects, for example, projects TR4…TR8 were conducted with one single company as part of a multi-year research program which included multiple projects. Also, as the time durations shows, some of the projects were overlapping in the same time horizon.

To quantitatively assess the success of projects from both industrial and academic standpoints, we used a 5-point Likert scale: (N/A): Not applicable; (0): Failure; (1): Some success; (2): Average success level; (3): Successful; and (4): Very successful. The industrial success criterion was the extent to which the developed approach benefitted the industry partner and was utilized in practice. The industrial success perception of a given project was also considered higher if it helped the industrialists get a better view on their challenges or to better characterize their SE practices. The academic success criteria were: number and quality of papers published from the project, and the number of young researchers (graduate students) trained in the project. A few other factors were also considered in assessing each of the above two success levels, e.g., one of the projects had reasonable success and progress, but it was terminated in the middle due to management changes in the industrial partner. For confidentiality purposes and sensitivity of such issues, we are not reporting some of such details. Of course, in cases like the above example, the industrial success level was *hurt* slightly and it also impacted academic success level of the project, e.g., in the above case, the graduate students no longer had access to the industry data and systems and we had to help them finalize their studies based on often partial data.

Author #1 or #2 were a key member of each project and they assigned the above measures, by reviewing their diary reflections and how each project impacted SE practice in industry partners. To prevent bias, and to ensure precision and consistency in the success measures, the authors applied "triangulation" by discussing with other members of each project (when possible) and also with each other about the above criteria and the success score of each project. In total, feedback from more than 10 people were involved in assessing the success measures of the projects. We also provide the number of young researchers trained in each project in the online supplementary document [13], which was used a factor for assessing academic success. It is our hope that by providing our beliefs and assessment criteria (as "observers"), readers can interpret our observations about success levels of the project. After all, the success of an IAC project is a multi-faceted metric and not a trivial aspect to be easily measured, but we believe we did our best to use the data and heuristics in our disposal to assess them.

Collaboration mode is a rubric in range of [1, 5] which we have adopted from the work of Wohlin [14], and we will discuss in the sections below.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

Table 1- Pool of the IAC projects analyzed in this study

Project ID	Authors involvement			Duration	Collaboration mode	Industry partner's…		Need (challenge)	Solution	Industry impact	Assessment of success	
	1	2	3			Domain	Size				Industrial success	Academic success
CA1-performance-testing	x			2003-2007	3	Real-time software	L	Model-based performance testing of distributed real-time software	A UML-based performance testing approach was developed in academia based on a general view of industry needs.	Since industry connection was "remote" (collaboration level=3), effort was made to "transfer" the technology in industry, but with no success	None	Very successful
CA2-oil-optimization	x			2007-2012	4	Oil industry	L	Need for optimization software for optimization the pumping cost of oil pipelines	An optimization software was developed to reduce the pumping cost of oil pipelines	Based on quantitative case study, the optimization software was able to reduce the pumping cost of oil pipelines.	Successful	Very successful
CA3-embedded-testing	x			2009-2011	4	Embedded software	L	Need for more effective tool-support traceability analysis in embedded software	A traceability analysis tool-set was developed and released to the industrial context.	Based on results from improving case studies, the traceability analysis tool-set was found useful.	Some success	Successful
CA4-configuration-testing	x			2009-2010	5	Governmental enterprise system	M	Manual troubleshooting of environmental configuration issues and staging environment instability was tedious and error prone. For example, there were over 50 hours of service downtime in 3 months due to those issues.	An automated environment configuration testing was developed and released to the industrial context.	The staging environment instability issues were automated detected by the tool and corrected in minutes. The service downtime reduced to 0 - 10 minutes per week.	Very successful	Very successful
CA5-blackbox-testing	x			2009-2011	5	Control systems	S	Cost of manual testing was too high and too many regression faults were observed	A tool named AutoBBUT for automated test code generation for black-box unit testing was developed and released to the industrial context.	Based on results from improving case studies, with the help of the tool, about 46 hours of testers time was saved in each unit testing iteration.	Successful	Successful
CA6-communication-testing	x			2012-2013	5	Control systems	L	Automated software testing of communication frameworks	None (since the project was aborted in early execution phase due to data confidentiality issues)	None. Project got cancelled after planning.	None	None
CA7-when-to-automate	x			2012-2013	5	Control systems	L	Lack of a systematic approach to decide what test cases to automate in software testing	A systematic approach, based on optimization and system dynamics, was developed and released to the industrial context	Based on quantitative measurements, the approach improved the cost effectiveness of software testing activities.	Very successful	Very successful
CA8-documentation	x			2009-2012	5	Embedded software	L	Huge amounts of effort had been spent for developing and maintaining software documentation	A systematic approach and tool for optimizing cost of software documentation was developed and released to the industrial context.	Based on quantitative measurements, the approach optimized cost of developing and maintaining software documentation.	Average success	Successful
TR1-performance-testing	x			2017	5	Governmental enterprise system	L	A major web application had performance issues with high user loads	Systematic software performance testing practices were applied to improve the system	The performance of the web application improved and there was no problems with high user loads	Very successful	None

Project ID	Authors involvement			Duration	Collaboration mode	Industry partner's…		Need (challenge)	Solution	Industry impact	Assessment of success	
	1	2	3			Domain	Size				Industrial success	Academic success
TR2-GIS-testing	x			2014	5	Defense	M	The manual GUI testing of a family of safety-critical GIS software systems has proved to be very costly in the last several years	None (since the project was aborted in the planning stage due to sudden change in company management who no longer supported the project)	None	None	None
TR3-MDE	x			2013-2014	5	Defense	L	Testing and maintenance of safety-critical middleware communication protocols had various challenges, i.e., unsynchronized interface artifacts across various SDLC phases, inefficient maintenance and documentation	A model-driven engineering (MDE)-based approached was developed	Many of the challenges were addressed and improving case studies are now underway to quantitatively measure the benefits and highlight the areas for further improvements	Successful	Average success
TR4-test-maturity	x			2015-2016	5	Defense	L	Need for maturity assessment and improvement of test process and practices (using TMMI and TPI)	An empirical study was conducted to assess test maturity using TMMI and TPI	Maturity assessment has provided objective awareness about maturity of test practices. Improvement activities on test maturity was conducted and provided measurable benefits in the company.	Successful	Average success
TR5-GUI-testing	x		x	2015-2016	5	Defense	L	Need for improvement of test automation practices and tools	An empirical study was conducted to compare automated visual GUI testing tools.	The empirical study has enabled the company to better plan test automation activities.	Successful	Average success
TR6-simulation-testing	x		x	2016-2017	5	Defense	L	Need for development of a specific test automation framework	A test automation framework for testing helicopter simulation software was developed.	The test framework have provided measurable benefits in the company.	Very successful	Average success
TR7-defect-reports	x		x	2016	5	Defense	L	Assessing and improving the quality of software defect reports	A quantitative approach for assessing the quality of software defect reports was developed and applied.	Improvements have provided measurable benefits in the company	Very successful	Successful
TR8-test-strategy	x		x	2016-2017	5	Telecom	L	Need for a systematic test automation strategy	A systematic test automation strategy was developed and applied.	The test automation strategy has provided measurable benefits in the company	Successful	Successful
TR9-regression-testing	x			2015-2016	5	Defense	L	Need for a systematic regression test approach considering multiple cost / benefit criteria	A systematic regression test approach using genetic algorithms considering multiple cost / benefit criteria was developed and applied.	The regression test approach has provided measurable benefits in the company	Very successful	Successful
TR10-test-service-maturity	x		x	2016-2017	5	Defense and telecom (two partners)	L	Need for maturity assessment and improvement of test services for two test service providers	An empirical study was conducted to assess maturity of test services using "CMMI for Services".	Maturity assessment has provided objective awareness about maturity of test services. Improvement activities on test maturity are underway.	Very successful	Successful

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

Project ID	Authors involvement			Duration	Collaboration mode	Industry partner's…		Need (challenge)	Solution	Industry impact	Assessment of success	
	1	2	3			Domain	Size				Industrial success	Academic success
TR11-design-inspection	x			2016	5	Aviation	L	Need for automated inspection of software design documents	A tool for automated inspection of software design documents was developed	The tool has provided measurable benefits in the company	Successful	Successful
ABB1-finding-code-for-maintenance		x		2011-17	4	Multinational engineering conglomerate	L	Need for finding and understanding code relevant to a software maintenance task	A novel code search engine implemented in Visual Studio	Hundreds of internal company users, tens of thousands of external users have found the tool useful	Successful	Successful
ABB2-navigating-code-for-maintenance		x		2014-15	4	Engineering	L	Need for navigating and understanding code during software maintenance tasks	A program navigation tool implemented in Visual Studio	Hundreds of internal company users and a user study showing that the tool improved user's navigation behavior	Successful	Successful
ABB3-bottlenecks-in-bug-handling		x		2014-15	4	Engineering	L	Need for understanding bottlenecks and issues in bug handling processes	Ethnographic studies of bug handling and concrete recommendations re improving it	Improvements in workflow and tooling	Average success	Average success
ABB4-encouraging-best-practices		x		2015-17	4	Engineering	L	Need for encouraging best practices at an individual level for developers	A gamification plugin for Visual Studio that encouraged developers to adopt best practices	Measureable improvements in developer behavior	Successful	Successful
ABB5-reducing-interruptions		x		2015-18	4	Engineering	L	Need for reducing the amount or severity of interruptions for developers and other knowledge workers	Software that monitors busyness and automatically changes a physical light to show when one is interruptible or not	Measurable reduction in interruptions per day and positive user feedback	Very successful	Very successful
ABB6-programming-robots		x		2017-18	5	Engineering	L	Need for reducing the complexity of programming robots	App and new robot language aimed at making robots easier to program	Measurably better than the current state-of-the-practice in extensive user studies	Very successful	Successful
ABB7-analyzing-IDE-usage		x		2016-17	5	Engineering	L	Need for analyzing existing IDE usage data	Reports and approach for quickly analyzing usage data	Directly impacted product road map such that over 2 PY of potential wasted effort was avoided	Successful	Average success

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

4 EXPERIENCE AND LESSONS LEARNED

Based on our experience in the above projects, we synthesize and report next our experience and lessons learned. Our approach was to discuss among the three authors the best practices synthesized in [6], and then assess to see which ones have been the most effective in our projects. We did not intend to exhaustively list and discuss in this paper all our lessons learned, but instead we present next the most important findings as emerged from our discussions, together with concrete examples and reflections.

4.1 WORKING AS ONE TEAM

One of the most important aspects of IACs are their collaboration mode (style), or degree of closeness between industry and academia. One of the best models in this context is the one proposed by Wohlin [14], as shown in Figure 1. There are five levels in this model, which can also be seen as a "maturity" level: (1): not in touch, (2): hearsay, (3): sales pitch, (4): offline, and (5): one team. In levels 1-3, there is really no IAC, since researchers working in those levels only identify a general challenge of the industry, develop a solution (often with too much simplifications and not considering the context [15]), and then (if operating in level 3) approach the industry to try to convince them to try/adopt the solution. However, since such techniques are often developed with a lot of simplifications and assumptions which are often not valid in industry [2, 15], they often fail to be applicable in industry.

IAC really occurs in levels 4 and 5 of the model. The level 5 is the most ideal case in which both parties work as "one team" to identify a specific challenge, then to iteratively evaluate and validate draft solution approaches and finally deploy a tailored solution in the context of industrial partner.

In terms of modes of industry-academia closeness, 18 of the 26 projects in our pool were in level 5, 7 projects were in level 4 and one project (CA1) in level 3. In CA1, we identified a general challenge of model-based performance testing of distributed real-time software and a PhD study was conducted to develop a novel test approach. Industry and academia had only infrequent contacts during the approach development and, thus, the industrial applicability of the approach at the end was low, e.g., many assumptions and simplifications were made for the UML-based test generation approach. Such a research was referred to as "research-then-transfer" by Colin Potts in his 1993 IEEE Software paper [16]. Once we developed the test technique in CA1, we made the effort to "sell" the approach to several industry partners. It turned out that while a few practitioners "liked" the approach, no industry partner was ready to adopt it.

Based on the experience in CA1, the first author changed his research "paradigm" starting from CA2 to work in levels 4 or 5 of industry-academia closeness [14]. Although like any other collaboration, having industry and academia work in one team is not straight-forward and one could see a variety of challenges [6], but once there is mutual benefit and trust in both sides, such a collaboration could be fruitful, as can be seen in the list of projects. In context of his work in ABB, the second author and his colleagues have also often worked in mode 4 or 5.

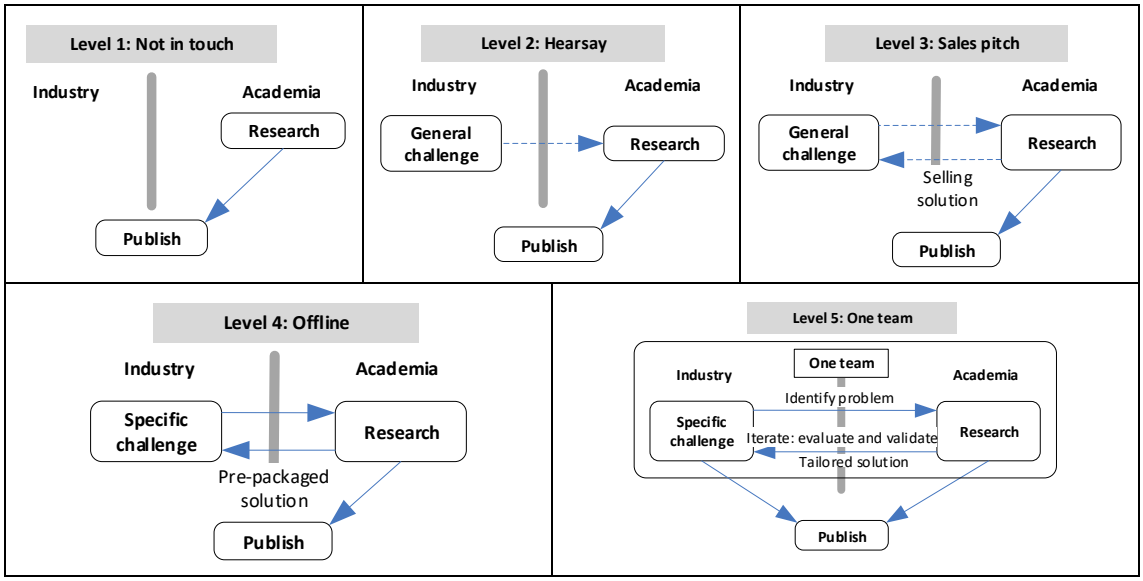

Figure 1- Five (maturity) levels of closeness between industry and academia (source: [14])

We have seen that working together leads to increased chances of industry "uptake" of the developed approaches. In all projects, except for CA1, the project outcomes were adopted by industry partners since the projects were rooted in the "real" industry problems in the first place, and solutions were developed together in iterations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

We believe other practitioners can benefit from our experience in utilizing the 5-step model of Wohlin, and we hope that it gives them more effective models for working with researchers.

4.2 IDENTIFYING THE (RESEARCH) PROBLEM

Identifying the "right" SE problem to work on is also very important. As Gordon Clegg mentioned in his 1969 class book "The design of design" [17]: "*Sometimes the problem is to discover what the problem is*".

Challenges (needs) of the industry partners drove the need for all the subject projects, and we used action-research [12] as our approach in all projects. In projects with working mode of 5, once both parties had the initial commitment to collaborate, they held a number of initial meetings to identify the "right" problem (industry need) for the collaboration. Various factors were considered in that activity, e.g., solving the problem should have been interesting w.r.t. academic literature and should have been beneficial to the industry partner. The first author worked with an industrial partner and presented a grounded-theory-based approach for selecting the "right" topics for IACs with high relevance and impact [18]. The topic-selection process involves interaction with company representatives in the form of both multiple group discussions and separate face-to-face meetings while utilizing grounded-theory to find (converge to) topics which would be 'interesting' and useful from both industrial and academic perspectives. For example, projects TR4… TR8 were in joint work with a large software company in Turkey and the approach [18] was systematically used to derive those project topics.

Others have also presented guidelines for topic selection in IACs, e.g., Misirli et al. [19] proposed four fitness criteria for this purpose: (1) concreteness of the topic, (2) suitability of the topic for experimentation, (3) relevance of the topic to research community, and (4) prior domain experience (of researchers). Item #3 above is of special importance. The topic to be chosen for an IAC shall have enough research "substance", and worthy enough of a research project. Otherwise, there is a "risk" that the project will be a "conventional" SE service or consulting. For example, the first author was approached by several countries in Canada and was asked whether he and his team could test a new software provide. He had to communicate with those companies that their objective In testing research was not just to conduct manual or automated testing, but find "innovative" ways worthy of research investigations.

Thus, we have proactively taken countermeasures in our research projects to avoid such risks. For example, in project CA8, a systematic approach and tool for optimizing cost of software documentation was developed and released to the industrial context. The approach was developed in a generalizable and extensible manner, which can be reused and extended by other researchers. In project TR10, we conducted an empirical study to assess maturity of test services using "CMMI for Services". The study approach could be again reused by other teams. Thus, those involved in IACs should ensure avoid generating results that do not generalize beyond the scope of the collaboration project.

As per the experience of the second author in conducting applied research in ABB, some of the ways used to identify the research problem at ABB have been: (1) Talking to the developers and developer management; (2) Monitoring developers anonymously (e.g., ethnographic studies, IDE instrumentation, etc.); (3) Having external consultants studying the context; and (4) Performing company-wide review of practices.

As another experience, the first author has held various meetings with many companies to determine the "right" problems in the area of testing. The researcher mentioned the wealth of literature on test-case design and that some of those techniques could be applied or extended for the industry partner, but most partners mentioned that they did not have challenges w.r.t. test-case design, but mainly had needs for improving efficiency of testing, mostly via test automation. This was an indication that most of the software testing literature, on test-case design, has not adequately considered its research "utility" and the actual need of the industry. Similar observations have been also reported in other studies, such as [3], which shared the experience of a researcher from applying software testing academic approaches in industry and limitations of many of those approaches in practice.

In terms of "who" takes the initiative for an IAC, a researcher may approach industry and say either: "I have developed a technique to improve SE, or have a topic in mind. Are you interested to collaborate?" or, "Let's choose a topic together and solve it". An industrialist may also approach a researcher and say either: "We have a challenge with [a SE topic]. Do you have the expertise to help us?". We have seen that, often times, industry is more receptive for the latter approach as practitioners prefer to be involved in the process of finding a problem in their own contexts.

In summary, identifying the "right" (research) problem is of outmost importance for an IAC. When talking about the issue of choosing research problems, we could also refer to a previous paper [20], which compared the industrial and academic focus areas in software testing, and found them to be "worlds apart". Such an observations in fact verifies our argument in the current paper that the chosen SE topic (problem) should obviously be industrially-relevant to attract and keep the industry's commitment and should also be a topic on which "rigorous" research can be done, and published.

4.3 ENSURING PRACTICALITY AND APPLICABILITY OF APPROACHES

Some critics say that: "*Software engineering research suffers from irrelevance. Research outputs have little relevance to software practice*" [1]. Many believe that one main reason for this syndrome is that often the approaches, presented in papers, are not

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

practical or applicable in practice [1], which in turn are due to various reasons, e.g., applying a SE technique has more costs for applying than the benefit to be gained [3], or the inputs needed by most SE techniques are not readily available in most projects [3]. This brings us to our next learnt lesson:

> ➢ For industry partners to adapt and utilize a SE technique in their work practices, it has to be practical and applicable in their contexts.

In all our projects practicality and applicability of approaches were carefully considered throughout the entire IAC. As an example, in CA8, a systematic approach and tool-set for optimizing cost of software documentation was developed and released to the industrial context. To ensure practicality and applicability of the approach, we ensured designing it in a way that would need inputs which are already available or can easily be extracted from the context. Also, if an approach works on an academic prototype system does not necessarily mean it will work in practice. The industry partner was actively involved to ensure the method would actually work in their context.

4.4 CONDUCTING COST-BENEFIT ANALYSIS OF APPROACHES

A researcher, who moved to industry, mentioned in his blog that: *"[SE] managers are coin operated in some sense. If you can't quantify it in terms of time or in terms of money, it doesn't make much difference to them"* (goo.gl/aj55u)

In an IEEE software paper entitled "*Practicing what we preach*" [21], the author argued that "*any modeling or formalizing [in SE research] would have to be lightweight*". Another paper [22] argued that "*lack of concrete knowledge of what organisations can gain from applying state-of-the-art but also time-consuming [SE] approaches is one of the major obstacles in getting research results adopted by practitioners*".

> ➢ To convince industry partners to adapt and utilize a SE technique in their work practices, researcher should conduct cost-benefit analysis (as quantitative as possible). Industry will only adopt the approach if the associated benefits outweigh the costs.

We discuss two cost-benefit analysis examples from our projects (ABB6 and CA5) next. In ABB6, when we were considering CoBlox, an easier user-interface for programming robots, we faced a complex cost-benefit analysis scenario. On the one hand, making our robots easier to program was an obvious benefit. On the other hand, at the start of the project there were seemingly many potential pitfalls to adding a new programming interface to the existing ecosystem in the context of the ABB. How would the new interface interact with the existing system? How would experts and beginners work together? How would existing smart commands be integrated? We determined that enabling experts to tweak and improve the beginner-generated code was the most-important negative to address. By identifying and eliminating this key business concern via a technical solution early on, our cost-benefit analysis became much simpler at the end, and thus we implemented the solution.

As another example, in our project CA5, we conducted quantitative cost-benefit analysis of our black-box automated testing by empirically measuring its ROI (Return On Investment) as follows:

ROI =Benefit driver (BD) 1+ BD2-Cost driver (CD) 1-CD2 =
87 hours (initial development of manual test suite, if it was to be done)
+87*6 hours (test code maintenance, since the system evolved 6 times)
-120 hours (the test tool's development time)-3 hours (test code inspection and completion)
ROI = 486 hours = 60 man-days

Calculating the ROI of the above particular testing approach helped both the practitioners and academics involved in the project to objectively assess the benefit of the approach. Note that hidden costs such as acquiring relevant knowledge are not considered in the above calculation, since we assumed that the engineers already have the relevant knowledge. In fact, MSc students who were already trained with test automation and the approach conducted the work, thus that cost was negligible. The industrial partner was glad that the ROI was positive and the approach was consequently put into active use in the company.

In summary, from the study and analysis that we have done from the 26 projects mentioned in Table 1, we have observed a direct relationship between industrial success of a project and the cost-benefit analysis carried out. For majority of our projects, when possible, a quantitative or qualitative ROI analysis was carried out (like the above example).

4.5 NEED FOR MATURITY OF RESEARCH PROTOTYPE TOOLS

When a given project needs a research prototype tool to apply the method in practice, it is often the case that researchers put only limited effort to build simple tools with low usability. Of course, we have learnt that maturity of such prototype tools depends on goal of a given project and usage scale of the tool. For example, the researchers in ABB often use the Technology Readiness Levels (TRL) (goo.gl/1pAQtU) for assessing about how mature they want their prototypes tools to be, and to find the right balance between a commercially-polished tool and a research prototype. We discuss a few example projects below.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2914663, IEEE Software

In the project ABB1 which focused on finding code relevant for a software maintenance task maintenance, a tool named *Sando* was developed. As judged by its users, it had a "nice UI" but only worked in the "happy cases". Thus, it was acceptable for research purposes, and for use in circumstances where code search was especially important, but it had limited commercial usage inside and outside the company. Note the even with these limits to its adoption, *Sando* was used by hundreds of ABB users and was downloaded by over fifty thousand external users.

In a similar project ABB2, focusing on navigating-code-for-maintenance, another program navigation tool named *Prodet* was implemented in Visual Studio which had acceptable maturity for its intended usage, i.e., limited user-base in the company. It was used by hundreds of internal ABB users but was not released to external users.

In the project ABB5, a software named *FlowLight* for monitoring busyness of developers was developed which can automatically control a physical light to show when a developer is interruptible or not. The system has also recently been commercialized (www.embrava.com/pages/flow). The early versions of the software had a simple but robust GUI which was acceptable for its initial academic evaluations. Of course, the latest commercial versions have improved a lot compared to the initial prototype versions.

In CA5, a tool named AutoBBUT for automated test code generation for black-box unit testing was developed. The tool was utilized in the industrial context, and to encourage further adoption, it was released as an open-source tool.

4.6 Encouraging further adoptions

In an IAC, the proposed methods are often only used by the industry partner(s) involved in the project, thus there is a need to encourage further adoptions in academia and other industrial contexts. In the experience of all three authors, we have observed that various initiatives can help when it comes to encouraging further adoptions, e.g., giving webinars and talks (in industrial conferences or user groups) about the technique and its usefulness, convincing management of other companies, encouraging usage by management "decree" (top-down decisions), and publishing in trade magazines.

Projects CA7 and TR10 are two examples. The project CA7 was conducted in collaboration with a Canadian company and had a successful outcome. The goal was to develop a systematic approach to decide what test cases to automate in software testing. When the first author moved to Turkey, he had discussions with several companies and, by convincing their test teams, he applied the approach with some improvements in their contexts too. Similarly, the method used in project TR10 was first developed for two companies. The goal was maturity assessment and maturity improvement of test services. Afterwards, the need was observed in the context of one other company, and the approach was re-applied in that context too.

4.7 Long-term benefits and benefits beyond the involved parties

In our projects, we have observed "long-term" benefits for the involved parties, beyond just the outcomes and benefits made in one single IAC project.

We also noticed a *culture change* (or even "shift") for several of our partners. By seeing how collaboration with researchers can provide value to them, several of our partners showed interest to initiate further follow-up IACs with the researchers' team. Also several partners (in the case of CA3, CA4, TR6 … TR8) started to find more interest in academic literature in SE and have started to form sub-groups inside their companies to read and discuss empirical SE studies, and have considered applying existing methods in their SE practices. Thus, for the case of most of the partners, the projects have provided sustaining benefits (both tangible and intangible) beyond just benefits in one single IAC project. Also, we did our best to nurture our relationships, in a few cases (e.g., for CA4-configuration-testing, TR6-simulation-testing), we contacted our partners a few years after the collaborations had ended to hear about the latest developments and evidence w.r.t. the results and benefits of the projects. Feedbacks that we have received so far have indicated positive evidence about those aspects even a few years after the IACs.

Let us discuss project TR6 (simulation-testing) and TR7 (defect-reports) as examples. In TR6, a test automation framework for testing helicopter simulation software was developed. The test framework provided measurable benefits in the company. The long-term impact and benefits of the project were two-fold: (1) the teams in the learned how to conducted quantitative measurements to assess cost-effectiveness of test automation; (2) the SE best practices involved in designing the test automation framework (such as design and test patterns) increased the know-how and knowledge base in the company, and plans were made to develop other frameworks for other test purposes and other projects. We have even heard from our partners that results and approaches in a few projects (such as TR6) have been reused in other follow-up projects or other engineering units of the companies. For example, the reader is referred to a recent IEEE Software paper [23], related to project TR6.

In TR7, a quantitative approach for assessing the quality of software defect reports was developed and applied. This project too had long-term benefits, beyond just the project itself, e.g., (1) ideas from the work were planned to be utilized for assessing and improving the quality of software documentation in general in company. The company had CMMI Level-3

certifications, so documentation was an important issue; and (2) there was a clear change in "mentality" of the involved software engineers as they increased their know-how's on to assess SE improvements more "rigorously".

There are also long-term benefits for the researchers involved, since as they see the evidence that IAC could be valuable, they would be interested to initiate further IACs in the long-term future.

We have also observed benefits "beyond" the involved parties:

- Benefits for the practitioners' community: Since the approaches developed in our IAC projects have been assessed in industrial contexts, other practitioners could review and benefit from them. Studies [24] are showing that practitioners prefer the evidence, which has been assessed in industrial contexts and their peer opinions rather than pure academic papers. We have, for example, presented the outputs and experience from our projects as presentations in many industry conferences, such as in Test Automation Days 2018 conference in the Netherlands (www.testautomationday.com/test-automation-day-2018). In fact, in a few occasions, our experience and "portfolio" in specific projects (such as TR6-simulation-testing and TR9-regression-testing) has provided additional concrete benefits by having other companies approach us and asking us to apply the same or similar approaches (e.g.., test automation) in their contexts, to initiate new research partnerships and even consulting projects. Thus, for sure, we have observed many benefits "beyond" the involved parties from our projects. Also, in a few other cases, evidence and "peer opinions" of practitioners in our projects [24] have been communicated with other companies without us being in the loop initially, and those new partners have then approached us for joint collaborations on similar topics. In summary, this has clearly shown to us that fruitful IACs would lead to a wider network of partners and more follow-up collaborations (like a "virtuous cycle").
- Benefits for the research community: In most cases, again when the IAC is planned and executed properly, results could be generalized beyond just the involved parties. For example, in projects CA7 (when-to-automate) and CA1 (performance-testing), we developed search-based approaches (based on genetic algorithms) which has been cited and used by other researchers. In the papers that we have published out of the projects, we have generalized the needs of the industry and the solutions that we developed. Also, when other researchers see the evidence and hear the experience that IAC can be successful, we have seen that they would attempt to conduct (more) IAC research. This, in a sense, becomes a "virtuous cycle" for the SE community.

5 Conclusions and Road Ahead

In conclusion, the authors believe that, for the SE research community to have a meaningful future, there is a critical need for more connections and collaborations between industry and academia. We have found that IACs are an effective vehicle for conducting research with industry impact, yet that not all IACs result in industry impact due to common challenges and mis-steps (as summarized in the SLR [6]). We have summarized our lessons learned, presented above, to increase the chances of successful and impactful research from IACs. The lessons 1 and 2 above aim at making IAC more successful; lessons 3, 4 and 5 are about improving the chances of industrial impact; and lessons 6 and 7 aims at expanding IACs.

We would like to provide the following recommendations to practitioners, which could lead to better connections between industry and academia:

- When you face technical challenges, first search for the relevant academic research about your problem, and do not try to develop a solution from scratch on your own. The solution(s) that you would develop on your own would not, in general, be necessarily the best ones. Such suggestions for dialogue between researchers and practitioners have also been reported elsewhere [25].
- Report your challenges and experiences as blog posts and papers and let researchers know about them. Although you may not think so, they are often valuable in letting researchers know about practical challenges and then work on them.

Successful SE research is best done when there is both high methodological rigor and high practical relevance. Only one is not enough. As it has been also widely discussed in other fields [26], this is often referred to as "pragmatic" science, in contrast to "pedantic" science (high rigor, low relevance), "popular (popularist)" science (low rigor, high relevance), or "puerile" science (low rigor, low relevance). In a 1994 IEEE Software paper [27], Robert Glass mentioned that, in his opinion, the SE research was in "crisis", since most research activities at the time were not (directly) relevant to practice. We wonder how much the situation has changed in the last 25 years since 1994.

Industry-relevant research and making an impact in the software industry involves more than just producing research results and disseminating them in publications which are, honestly, not usually read by practitioners. Conducting industry-relevant research requires close cooperation and collaboration between industry and academia throughout the entire research process, from the identification of the problem all the way to delivering the result and publications. To ensure win-win for both sides, they need to follow lean collaboration models and success criteria briefly discussed in this article, including but not limited to the need for continuous cost-benefit analysis of joint R&D efforts.

REFERENCES

[1] A. Tang and R. Kazman, "On the Worthiness of Software Engineering Research," *Technical report,* http://shidler.hawaii.edu/sites/shidler.hawaii.edu/files/users/kazman/se_research_worthiness.pdf, 2017.

[2] A. Yamashita, "Integration of SE Research and Industry: Reflections, Theories and Illustrative Example," in *IEEE/ACM International Workshop on Software Engineering Research and Industrial Practice*, 2015, pp. 11-17.

[3] A. Arcuri, "An experience report on applying software testing academic results in industry: we need usable automated test generation," *Empirical Software Engineering,* vol. 23, no. 4, pp. 1959-1981, 2018.

[4] R. M. C. Andrade, V. Lelli, R. N. S. Castro, and I. S. Santos, "Fifteen Years of Industry and Academia Partnership: Lessons Learned from a Brazilian Research Group," in *International Workshop on Software Engineering Research and Industrial Practice*, 2017, pp. 10-16.

[5] D. Karanatsiou, Y. Li, E.-M. Arvanitou, N. Misirlis, and W. E. Wong, "A bibliometric assessment of software engineering scholars and institutions (2010–2017)," *Journal of Systems and Software,* vol. 147, pp. 246-261, 2019.

[6] V. Garousi, K. Petersen, and B. Özkan, "Challenges and best practices in industry-academia collaborations in software engineering: a systematic literature review," *Information and Software Technology,* vol. 79, pp. 106–127, 2016.

[7] V. Garousi, M. M. Eskandar, and K. Herkiloğlu, "Industry-academia collaborations in software testing: experience and success stories from Canada and Turkey," *Software Quality Journal,* vol. 25, no. 4, pp. 1091–1143, 2017.

[8] V. Garousi, M. Felderer, J. M. Fernandes, D. Pfahl, and M. V. Mantyla, "Industry-academia collaborations in software engineering: An empirical analysis of challenges, patterns and anti-patterns in research projects," in *Proceedings of International Conference on Evaluation and Assessment in Software Engineering*, Karlskrona, Sweden, 2017, pp. 224-229.

[9] R. M. Emerson, R. I. Fretz, and L. L. Shaw, "Participant observation and fieldnotes," *Handbook of ethnography,* pp. 352-368, 2001.

[10] D. E. Knuth, "The errors of tex," *Software: Practice and Experience,* vol. 19, no. 7, pp. 607-685, 1989.

[11] A. J. Ko, "A three-year participant observation of software startup software evolution," in *Proceedings of International Conference on Software Engineering: Software Engineering in Practice Track*, 2017, pp. 3-12.

[12] D. S. Cruzes and T. Dyba, "Synthesizing evidence in software engineering research," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010, pp. 1-10, 1852788.

[13] V. Garousi, D. Shepherd, and K. Herkiloğlu, "Supplementary material for the IEEE Software paper-Engagement of practitioners and SE researchers," http://doi.org/10.5281/zenodo.2527230, Last accessed: April 2019.

[14] C. Wohlin, "Software Engineering Research under the Lamppost," in *Proceedings of the International Joint Conference on Software Technologies*, 2013.

[15] L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "The Case for Context-Driven Software Engineering Research," *IEEE Software,* vol. 34, no. 5, pp. 72-75, 2017.

[16] C. Potts, "Software-engineering research revisited," *IEEE Software,* vol. 10, no. 5, pp. 19-28, 1993.

[17] G. Glegg, *The Design of Design*. Cambridge University Press, 1969.

[18] V. Garousi and K. Herkiloğlu, "Selecting the right topics for industry-academia collaborations in software testing: an experience report," in *IEEE International Conference on Software Testing, Verification, and Validation*, 2016, pp. 213-222.

[19] A. T. Misirli, H. Erdogmus, N. Juristo, and O. Dieste, "Topic selection in industry experiments," presented at the Proceedings of the International Workshop on Conducting Empirical Studies in Industry, Hyderabad, India, 2014.

[20] V. Garousi and M. Felderer, "Worlds apart: industrial and academic focus areas in software testing," *IEEE Software,* vol. 34, no. 5, pp. 38-45, 2017.

[21] S. Ghaisas, "Practicing what we preach," *IEEE software,* no. 1, pp. 88-92, 2014.

[22] H. Kaindl *et al.*, "Requirements engineering and technology transfer: obstacles, incentives and improvement agenda," *Requirements Engineering,* vol. 7, no. 3, pp. 113-123, 2002.

[23] V. Garousi *et al.*, "Experience in automated testing of simulation software in the aviation industry," *IEEE Software,* July/August 2019.

[24] P. Devanbu, T. Zimmermann, and C. Bird, "Belief and Evidence: How Software Engineers Form Their Opinions," *IEEE Software,* vol. 35, no. 6, pp. 72-76, 2018.

[25] C. L. Goues, C. Jaspan, I. Ozkaya, M. Shaw, and K. T. Stolee, "Bridging the Gap: From Research to Practical Advice," *IEEE Software,* vol. 35, no. 5, pp. 50-57, 2018.

[26] N. Anderson, P. Herriot, and P. Hodgkinson Gerard, "The practitioner-researcher divide in Industrial, Work and Organizational (IWO) psychology: Where are we now, and where do we go from here?," *Journal of Occupational and Organizational Psychology,* vol. 74, no. 4, pp. 391-411, 2010.

[27] R. L. Glass, "The software-research crisis," *IEEE Software,* vol. 11, no. 6, pp. 42-47, 1994.

AUTHOR BIOGRAPHIES

Vahid Garousi is an associate professor at Wageningen University. His research interests include software testing, empirical software engineering, and software engineering in practice. He also actively provides consulting and corporate training to software companies. He received a PhD in software engineering from Carleton University. He was selected as a Distinguished Speaker for the IEEE Computer Society from 2012 to 2015. Contact him at vgarousi@gmail.com

David C. Shepherd is a Senior Principal Scientist in ABB Corporate Research, USA. His research interests include software engineering, software evolution and reengineering. He received a PhD in Computer Science from University of Delaware. Contact him at davidshepherd@gmail.com

Kadir Herkiloğlu is the manager of test engineering group at HAVELSAN. His areas of expertise include software testing in the following domains: simulation and training systems; command and control systems for air, ground, and naval platforms. He received an MSc in telecommunication engineering from Istanbul Technical University. Contact him at kherkiloglu@havelsan.com.tr