Editor: **Jeffrey C. Carver**
University of Alabama
carver@cs.ua.edu

Collaborations and Code Reviews

Jeffrey C. Carver, Bora Caglayan, Mayy Habayeb, Birgit Penzenstadler, and Aiko Yamashita

Practitioner's Digest aims to bring practitioners into contact with research that might be of interest. Drawing on a team of correspondents, the column will highlight industry-relevant research results from software-engineering conferences. This information will appear in two formats. First, the column will provide short, one- to two-paragraph, synopses of papers presented at recent conferences. Second, from time to time, the column will present longer summaries of important articles. Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices included in the column, please send me and the paper authors a note about your experiences. —*Jeffrey C. Carver*

THE INTERNATIONAL Conference on Software Engineering (ICSE) is one of the premier annual events in the software-engineering research community. Besides the main conference, there are several satellite workshops and smaller conferences. This inaugural edition of Practitioner's Digest reports on two key industry-relevant themes that emerged during these events at ICSE '15: industry–academic collaborations and modern code review.

Industry–Academic Collaborations

This column aims to bring relevant research results to interested industry practitioners. So, it seemed appropriate to start it with two papers on effective industry–academic collaboration.

"Principles and a Process for Successful Industry Cooperation—the Case of TUM and Munich Re," by Maximilian Junker and his colleagues, described the success factors behind a long-term collaboration between Technische Universität München (TUM) and the reinsurance company Munich Re. This collaboration has facilitated eight projects focused on the quality of software development artifacts. It has been beneficial both to Munich Re (in terms of improved software-engineering methods) and TUM (in terms of two spin-off companies and numerous research publications).

Retrospective analysis determined that the collaboration's success resulted from following 12 key project management principles grouped into four categories. The *project content* principles are to select relevant problems, select manageable-sized problems, and choose concrete, non-invasive solutions. The *staffing* principles are to include industry staff in project teams, involve the problem owners, and involve management. The *organization* principles are to maintain regular meetings and disseminate project results. The *mindset* principles are to allow creative leeway, enable trustful interaction, be open to criticism, and appreciate mutual interests. Although these principles might lead to general nodding and the question "So what?," the paper compellingly explained what those principles mean, why they're important, and the research process that helps realize them. This paper was part of the ICSE '15 Software Engineering Research and In-

0740-7459/15/$31.00 © 2015 IEEE

dustrial Practice Workshop and received the *IEEE Software* Best Paper Award (see p. 4.); access it at http://goo.gl/hK1yCk.

"Fast Feedback Cycles in Empirical Software Engineering Research," by Antonio Vetrò and his colleagues, described the ongoing discussion on Empirical Software Engineering 2.0 as a way to improve empirical-research results' impact on industrial practice. A fast feedback cycle is enabled by first identifying key concepts that need studying and then designing and quickly executing a small proof-of-concept study to demonstrate the approach's potential benefits, by leveraging the use of interactive data analysis and mining.

In an illustrative example, Vetrò and his colleagues worked with Scrum-based projects to develop and evaluate a mechanism that analyzes user stories to discover scope creeps regarding project goals. The authors used their model to infer topics and then received quick feedback from the industrial partner. This fast feedback let the researchers successfully tune the extraction model.

One study participant said, "I found it interesting to see, in the pilot study, how clustering the words of our user stories pointed out what underlying interdependencies we have in our requirements. We are now applying the same principles of the iterative feedback methodology to improve our processes, for example, concerning estimation quality and semantic clarity of requirement specifications."

This research has two implications for fast feedback cycles:

- Fast iterations of feedback can increase a model's precision and the results' value.
- Empirical evaluation can be more effectively integrated during live development, benefitting both practitioners (through more current feedback) and researchers (through more accurate and real-time data).

This paper was part of the ICSE '15 New Ideas and Emerging Results track; access it at http://goo.gl/HKP0VD.

Modern Code Review

Modern code review, a lightweight version of traditional code inspection, has been increasing in relevance and frequency in the research and industrial communities. During the events connected to ICSE '15, three papers discussed aspects of modern code review.

"Investigating Code Review Practices in Defective Files: An Empirical Study of the Qt System," by Patanamon Thongtanunam and her colleagues, examined 11,736 reviews of changes in 24,486 files to investigate how the characteristics of code review practices impacted the resulting software's quality. The paper included these key findings:

- Reviewers aren't as careful when they review changes made to risky files (files that have been defective in the past). These risky files often have defects in the future.
- Issues addressed during code reviews tended to focus on easing future maintenance rather than identifying specific functional problems.
- The metric with the largest relative impact was the number of authors, whereas the proportion of revisions without feedback and the review length metrics had the smallest impact.

Besides these results, you might be interested in the analysis methodology and full list of metrics for evaluating code review effectiveness. This paper was part of the 12th Working Conference on Mining Software Repositories; access it at http://goo.gl/XSOIVO.

"Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets," by Mike Barnett and his colleagues, addressed the problem of noncohesive changesets (sets of files committed to a repository as a change unit). Such changesets occur when developers include unrelated changes in the same commit. (For example, while fixing a defect, a developer improves readability by renaming variables.) This dilution of focus increases the review process's difficulty.

Barnett and his colleagues reported that on a set of 1,000 reviews (randomly sampled from changes in two Microsoft products), more

Fast iterations of feedback can increase a model's precision and the results' value.

than 40 percent of the changes were decomposable into multiple partitions. Microsoft researchers developed ClusterChanges, a tool that automatically partitions noncohesive changesets to let reviewers examine the changes separately. The 20 developers who participated in a user study found the automatic decomposition useful. According to one user, "[Decomposing changes] is useful because it allows different reviewers with different purposes to focus on what they want." ClusterChanges isn't tied to a specific development environment, programming language, or application domain, making it potentially useful in various situations. This paper appeared in the main research track of ICSE '15; access it at http://goo.gl/nl82n4.

"Code Reviews Do Not Find Bugs. How the Current Code Review Best Practice Slows Us Down," by Jacek Czerwonka and his colleagues, analyzed the use of code reviews and their effects on software quality. The analysis was based on data from 25,000 Microsoft developers who performed code review using the CodeFlow tool, which was used in at least 60,000 sessions per day.

One interesting observation related to the code reviews' goal. Defect identification, typically assumed to be the primary goal, accounted for only 15 percent of the code review comments. Improving the code's maintainability accounted for more than 50 percent of the code review comments.

Another interesting observation related to the number of files to include in a changeset for review. The rate of usefulness of review comments for changesets of up to 20 files was steady. Beyond that saturation point, the average comments' usefulness decreased.

In addition, the results showed that the developers needed 6 to 12 months of experience with the code to be effective reviewers. In some cases, code authors added reviewers because they didn't want to exclude team members; this behavior could slow down the review process.

Finally, it was necessary to evaluate the comments' severity and allow only the severest comments to block a commit's acceptance. Overall, the results indicated that developers couldn't ignore code review's social aspects if they wanted the reviews to be successful. This paper was part of the ICSE '15 Software Engineering in Practice track; access it at http://goo.gl/dh6UR3.

The next edition of this department will report on the remaining industry-relevant papers from the workshops and conferences connected to ICSE '15.

JEFFREY C. CARVER is an associate professor in the University of Alabama's Department of Computer Science. Contact him at carver@cs.ua.edu.

BORA CAGLAYAN is a postdoc in Ryerson University's Department of Mechanical and Industrial Engineering. Contact him at bora.caglayan@ryerson.ca.

MAYY HABAYEB is a master's student in Ryerson University's Data Science Laboratory. Contact her at mayy.habayeb@ryerson.ca.

BIRGIT PENZENSTADLER is an assistant professor of software engineering at California State University, Long Beach. Contact her at birgit.penzenstadler@csulb.edu.

AIKO YAMASHITA is a data analyst and entrepreneur at Yamashita Research and is an adjunct associate professor at Oslo and Akershus University College of Applied Sciences. Contact her at aiko.fallas@gmail.com.