SPECIAL ISSUE ON REALIZING ARTIFICIAL INTELLIGENCE SYNERGIES IN SOFTWARE ENGINEERING

Çetin Meriçli (Carnegie Mellon University, USA), Burak Turhan (University of Oulu, Finland)

With its boundaries expanding into other disciplines and fields as the computers become more and more ubiquitous, Software Engineering (SE) is expected to solve a plethora of increasingly complex questions that are dynamic, automated, adaptive, or must execute at a very large scale. In theory, there could be another layer of collaboration between SE and other disciplines in addition to employing software systems of varying complexity. For example, Artificial Intelligence (AI) technologies can support the development of increasingly complex SE systems as in the case of recommendation systems. Conversely, in theory, SE might also play a role in alleviating development costs and the development effort associated with AI tools and applications such as robotics where proper development and testing practices are of utmost importance from both cost and robustness perspectives. Unfortunately, in practice, such collaborations between SE and AI are rarely achieved.

This special issue is the joint result of invited papers from RAISE'14 Workshop on Realizing Artificial Intelligence Synergies in Software Engineering and an open call for contributions to address the issues above. All submissions were reviewed by experts in the field. Finally, three papers were selected for inclusion in this special issue. These papers cover specific AI/SE synergies such as experiments with boosting based cost sensitive transfer learning for cross-project defect prediction, a systematic review on the applications of Bayesian networks in software quality estimation, and case studies in learning developer cues in the context of program comprehension.

We hope that these papers will be useful in reflecting the type of discussions in RAISE Workshop series, as well as in providing a view of the state of the art in the research areas covered by each paper. Before we provide brief summaries of the papers, we would like to extend our sincere thanks to the authors for their contributions, reviewers for their help and high quality feedback, and the editor in chief for making this special issue possible.

A transfer cost-sensitive boosting approach for cross-project defect prediction, Duksan Ryu, Jong-In Jang and Jongmoon Baik: This paper is an experimental benchmark study in cross-project defect prediction (CPDP). Authors propose a new learning technique called Transfer Cost Sensitive Boosting (TCSBoost), evaluate its performance on 15 datasets, and benchmark their results against the state of the art cross-project defect prediction techniques. Addressing class imbalance problem with SMOTE, TCBoost shows a significantly better performance than the state of the art approaches. The study also reports that using limited amounts of within project data does not improve the performance of TCBoost further. Authors conclude that their approach can help reduce the cost of software quality assurance activities.

A systematic literature review on the applications of Bayesian networks to predict software quality, Ayse Tosun, Ayse Basar Bener and Shirin Akbarinasaji: This paper reports a systematic literature review on the applications of Bayesian networks (BNs) to predict software quality. Motivated by the premise on the use of BNs in decision making, the review identifies 10 primary studies and evaluates dataset characteristics, techniques used for parameter learning and structure learning, use of tools and model validation techniques in these primary studies. Highlighting the shortcomings in reporting the applications of BNs in

software engineering domain, authors provide a set of guidelines for reporting the essential contextual and methodological details in reporting BN applications in software quality prediction to support further studies including replications.

Supporting comprehension of unfamiliar programs by modeling cues, Naveen Kulkarni and Vasudeva Varma: This paper reports two case studies on the use of cues for program comprehension. In the first case study, authors use standard comprehension tasks with the goal of identifying a set of cues that developers use to explore new programs. In the second case study, they investigate the effect of IDEs on the comprehension process. Then, the authors propose an approach to construct task-specific program summaries based on the identified cue choices. Their evaluation leads to the conclusion that their approach can be beneficial for novice developers when investigating unfamiliar programs by suggesting the right cues for them towards better program comprehension.