Software Qual J (2007) 15:237–240
DOI 10.1007/s11219-007-9015-6

Introduction to the special issue on: ''Software Quality Improvements and Estimations with Intelligence-based Methods''

Marek Reformat · Du Zhang

Published online: 10 May 2007

The IEEE definition of Software Engineering states that ''Software Engineering is (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software, and (2) the study of approaches as in (1)'' (IEEE Std 610.12-1990). This definition identifies software operation and maintenance processes to be as important as that of development, and translates into an increased emphasis on aspects of software engineering that ensure defect-free operations and easy modifications. All this is especially challenging at a time when both the size and complexity of software systems have increased dramatically.

Software engineers and project managers are striving to address various problems observed during the different phases of the software development life cycle. As a consequence, there is a growing interest in utilizing emerging intelligence-based techniques to support construction and analysis of software systems and data. Application of techniques such as machine learning, data mining, computational intelligence, and agent-based technologies to different aspects of software development and maintenance becomes an integral part of the application of advanced technologies to software engineering. Examples of already known applications are as follows: modeling software processes and products using fuzzy-based logic networks (Park, Lee, & Oh, 2003; Pedrycz, Breuer, & Pizzi, 2006), utilization of fuzzy reasoning for reverse engineering (Jahnke, 2004), automatic classification of program behavior using an active-learning paradigm with Markov models and clustering (Bowring, Rehg, & Harrold, 2004), ensemble-based classifier systems for modules' reliability (Lenic, Povalej, Kokol, & Cardoso, 2004), neural network for cost estimation (Tadayon, 2005), and automated software testing using agents (Dhavachelvan, Uma, & Venkatachalapathy, 2006) as well as evolutionary computing (Michael, McGraw,

M. Reformat (✉)
University of Alberta, Edmonton, Canada
e-mail: reform@ece.ualberta.ca

D. Zhang
California State University, Sacramento, USA

& Schatz, 2001; Miller, Reformat, & Zhang, 2006). Additional examples can be found in a *Software Quality Journal* article by Zhang and Tsai (2003).

Apart from the research effort in software development, industry is exhibiting a significant interest in improving the quality of software. Software engineers use a number of different tools to support quality-related activities and make them more efficient. Some of the applications of intelligence-based techniques in this area are software quality assessment using different machine learning algorithms (Lounis & Ait-Mehedine, 2004), estimation of module quality using decision trees (Reformat, Pedrycz, & Pizzi, 2003), and a variety of prediction models (Fenton & Neil, 1999, 2005; Khoshgoftaar & Seliya, 2003).

This special issue presents contributions from two special sessions on *Software Engineering with Computational Intelligence* at the 15th and 16th IEEE International Conferences on Tools with Artificial Intelligence, held in November 2003 in Sacramento and 2004 in Boca Raton, respectively. It includes five revised and extended versions of the best papers chosen from 12 papers presented at both special sessions, after a second round of peer review process.

This special issue demonstrates that intelligence-based approaches are successfully used in a variety of software quality related problems. Applications range from testing of complex systems, through dynamic software reconfiguration, interoperability of software components, to validation of on-line systems and module's quality prediction. Authors used a variety of techniques such as AI and temporal planning, agent-based technology, reinforcement learning, and semi-supervised learning. The papers are organized in the following sequence.

A process of system testing is targeted in the paper entitled ''Rapid Goal-Oriented Automated Software Testing using MEA-Graph Planning''. In this paper, Manish Gupta, Farokh Bastani, Latifur Khan, and I-Ling Yen evaluate the application of AI planning techniques in automated software testing. The key contributions of this paper include (1) a formal model of software systems from the perspective of software testing; and (2) a framework for an automated planning system for software testing. A case study is presented.

The quality of software systems depends on their optimal configurations. However, initial deployment and subsequent dynamic reconfiguration of a software system is difficult because of the interplay of many interdependent factors, including cost, time, application state, and system resources. As the size and complexity of software systems increase, procedures (manual or automated) that assume a static software architecture and environment are becoming untenable. Naveed Arshad, Dennis Heimbigner, and Alexander Wolf develop, in their paper ''Deployment and Dynamic Reconfiguration Planning for Distributed Software Systems'', a novel technique for carrying out the deployment and reconfiguration planning processes that leverages recent advances in the field of temporal planning. A case study applies the proposed approach to a system consisting of various components that communicate across an application level overlay network. The results are encouraging.

The third paper, ''Supporting High Interoperability of Components by Adopting an Agent-based Approach'' by Wenpin Jiao and Hong Mei, addresses issues of integrating components into systems. The authors target problems concerned with the interoperability of components due to the interaction mismatches at multiple levels, such as interaction behaviors between components and features imposed by architectural styles. They formalize components involved in different architectural styles using the pi-calculus, and study the formal foundation of the component interoperability.

The fourth paper deals with the issues of validation of biologically inspired soft computing paradigms. In ''Validating Neural Network-based Online Adaptive System: A Case

Study'', Yan Liu, Bojan Cukic, and Srikanth Gururajan look at online adaptive systems with the ability to cope with the demands of a changing environment. These changes induce uncertainty, which limit the applicability of conventional validation techniques to assure reliable performance. They discuss a dynamic approach to validate the adaptive system component. The proposed approach consists of two run-time techniques: (1) a statistical learning tool that detects unforeseen data; and (2) a reliability measure of the neural network output after it accommodates the environmental changes. A case study on NASA F-15 fight control system demonstrates effectiveness of the proposed technique.

''Software Quality Estimation with Limited Fault Data: A Semi-supervised Learning Perspective'' by Naeem Seliya and Taghi Khoshgoftaar is the last paper in this special issue. It addresses the important problem of software quality analysis when there is limited software fault or fault-proneness data. More specifically, the small set of modules with known fault-proneness labels is not sufficient for capturing the software quality trends of the project. The authors investigate semi-supervised learning with the Expectation Maximization algorithm for software quality estimation with limited fault-proneness data. Software data collected form a NASA software project is used during the semi-supervised learning process. The software quality model is evaluated with multiple test datasets collected from other NASA software projects.

We hope that the readers will enjoy the special issue and that they will benefit from the useful and encouraging results conveyed by the authors of the papers.

References

Bowring, J. F., Rehg, J. M., & Harrold, M. J. (2004). Active learning for automatic classification of software behavior. *ISSTA 2004—Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 195–205).

Dhavachelvan, P., Uma, G. V., & Venkatachalapathy, V. S. K. (2006). A new approach in development of distributed framework for automated software testing using agents. *Knowledge-Based Systems, 19*(4), 235–247.

Fenton, N. E., & Neil, M. (2005). A critique of software defect prediction models. In D. Zhang & J. J. P. Tsai (Eds), *Machine learning applications in software engineering* (pp. 72–86). World Scientific Publishing Co.

Fenton, N. E., & Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions of Software Engineering, 25*(5), 675–689.

IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology—Description.

Jahnke, J. H. (2004). Cognitive support in software reengineering based on generic fuzzy reasoning nets. *Fuzzy Sets and Systems, 145*(1), 3–27.

Khoshgoftaar, T. M., & Seliya, N. (2003). Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering, 8*, 255–283.

Lenic, M., Povalej, P., Kokol, P., & Cardoso, A. L. (2004). Using cellular automata to predict reliability of modules. In *Proceedings of the eighth IASTED international conference on software engineering and applications* (pp.781–785).

Lounis, H., & Ait-Mehedine, L. (2004). Machine-learning techniques for software product quality assessment. In *Proceedings of the fourth international conference on quality software, QSIC 2004* (pp. 102–109).

Michael, C. C., McGraw, G., & Schatz, M. (2001). Generating software test data by evolution. *IEEE Transactions on Software Engineering, 27*(12), 1085–1110.

Miller, J., Reformat, M., & Zhang, H. (2006). Automatic test data generation using genetic algorithm and program dependence graphs. *Information and Software Technology, 48*, 586–605.

Park, B.-J., Lee, D.-Y., & Oh, S.-K. (2003). Rule-based fuzzy polynomial neural networks in modeling software process data. *International Journal of Control, Automation and Systems, 1*(3), 321–331.

Pedrycz, W., Breuer, A., & Pizzi, N. J. (2006). Fuzzy adaptive logic networks as hybrid models of quantitative software engineering. *Intelligent Automation and Soft Computing, 12*(2), 189–209.

Reformat, M., Pedrycz, W., & Pizzi, N. J. (2003). Software quality analysis with the use of computational intelligence. *Information and Software Technology, 45*(7 SPEC.), 405–417.

Tadayon, N. (2005). Neural network approach for software cost estimation. *International conference on information technology: Coding and computing, ITCC 2.* (pp. 815–818).
Zhang, D., & Tsai, J. J. P. (2003). Machine learning and software engineering. *Software Quality Journal, 11*(2), 87–119.