Autom Softw Eng (2010) 17: 373–374
DOI 10.1007/s10515-010-0073-9

Editorial: data mining in software engineering

Robert J. Hall

Received: 6 July 2010 / Accepted: 6 July 2010 / Published online: 13 July 2010

Data mining, the automated or semi-automated extraction of useful knowledge from large poorly structured data sources, is a hot topic in all sorts of fields. Potential science applications include, for example, learning from the expected avalanche of data generated by CERN's Large Hadron Collider. Telecommunications companies apply data mining to detect fraudulent network usage. Companies in many areas of business apply data mining to improve their marketing and advertising. Law enforcement uses data mining to detect various financial crimes. Given the well known complexity of software engineering processes and artifacts, it is perhaps not surprising that data mining can be applied there as well.

Developing software intensive systems is difficult in large part because of the complexity of development processes and of the resulting artifacts. Understanding what is to be built and how to build it is a process of discovery driven by large amounts of data. There is experience data from past development projects; there is data derived from abstractions and models of requirements, e.g. by simulation; and there is data derived from the built artifacts themselves, such as code metrics. As the amount of such data grows, it becomes harder and harder for humans to manage it and glean useful and reliable knowledge from it. This points to a growing need for automated tools to help us with this software engineering problem.

One of the original driving visions of the ASE Community, perhaps the central one, as laid out in the original KBSA Report and other places, is to apply artificial intelligence techniques in building tools to deal with software engineering's complexity. This issue of *ASEJ* brings us three contributions very much in this spirit, applying machine learning to the problem of data mining for software engineering.

R.J. Hall (✉)
AT&T Labs Research, Florham Park, NJ 07932, USA
e-mail: Bob.ASEJ@gmail.com

- In "Defect prediction from static code features: current results, limitations, new approaches", Menzies, Milton, Turhan, Cukic, Jiang, and Bener use machine learning to predict which modules will likely contain the most defects, allowing testing managers to allocate precious testing resources accordingly. In particular, they show how tuning a meta-learning framework can outperform other ML approaches.
- In "Stable rankings for different effort models", Menzies, Jalali, Hihn, Baker, and Lum survey 158 different methods for predicting the likely effort required for a software development project. Their contribution is to isolate four of the 158 methods that among them outperform all the others; that is, for a given project in the case study, it is likely the best method will be one of the four and very unlikely that one of the other 154 will do well. Their case study demonstrates surprisingly robust stability of the results across a multidimensional parameter space defined by varying evaluation criteria, data sets, and random seeds. Clearly, if these results can be replicated and extended to other domains, this is a powerful and useful result indeed.
- Gay, Menzies, Davies, and Gundy-Burlet describe treatment learning systems and show in "Automatically finding the control variables for complex system behavior" that treatment learners can outperform traditional numerical optimization routines in isolating small sets of critical system parameters (ones on which a system's success or failure most sensitively depend). By isolating these critical sets, a development team can most efficiently allocate its costly resources like attention time of experts and testing resources. The paper's conclusions are supported by results from several NASA-derived case studies.

Please enjoy these interesting and significant contributions to the field of data mining for software engineering, and feel free to send your comments to the authors or to me.