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

Regression Testing, Spoken Language, Crash-Inducing Commits, UML, and Legal Policy

Jeffrey C. Carver, Jordi Cabot, Leandro L. Minku, and Marco Torchiano

THIS MONTH'S COLUMN reports on papers from the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, the 11th International Conference on Predictive Models and Data Analytics in Software Engineering, and the 2015 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices described in the column, please send Jeffrey Carver and the paper authors a note about your experiences.

Regression Testing

"Will This Bug-Fixing Change Break Regression Testing?," by Xinye Tang and his colleagues, describes a way to make regression testing less time-consuming and repetitive.[1] Using data extracted from the change history, including various types of metrics and whether a change led to a failed test, a machine-learning approach builds a predictive model to identify changes that might break regression testing. An evaluation of this approach on four large open source projects resulted in 66.7 to 83.3 percent precision (the percentage of changes identified as problematic that really were problematic) and 38.1 to 92.3 percent recall (the percentage of successfully identified problematic changes).

Once the approach identifies problematic changes, it uses static call graph analysis to identify which test cases could fail as a result of these changes. Developers can then focus on these test cases, fixing bugs before the costly execution of the full regression suite.

Tang and his colleagues' evaluation identified all failing test cases corresponding to the changes predicted as problematic before running the regression test suite. They achieved that result with an average of fewer than 20 test cases. So, developers only have to review a relatively small number of test cases to reduce the chance of the regression test suite failing. You can access this paper at http://goo.gl/UC8fCz.

Language Policies

"Language Matters," by Yi Wang, reports results of action research on the impact of an English-only policy in a Chinese outsourcing company.[2] Because English has become the lingua franca in global software development, companies might consider adopting English-only policies for teamwork and communication. At the studied company, all workplace communication had to be in English, including workplace conversations, email, chats, and meeting notes. In addition, all documents, including internal documents, had to be in English. The company enforced this policy company-wide for 50 days before reverting to the original policy (workplace communication could be in Chinese).

Wang collected data (through biweekly questionnaires and interviews) starting from before the English-only policy was launched until after it was removed. Analysis showed that the policy

- decreased work satisfaction, which increased quickly after the policy was removed;
- decreased teamwork quality (which included communication, coordination, balance of member contributions, mutual support, effort, and cohesion), which slowly started increasing after the policy was removed; and

0740-7459/16/$33.00 © 2016 IEEE

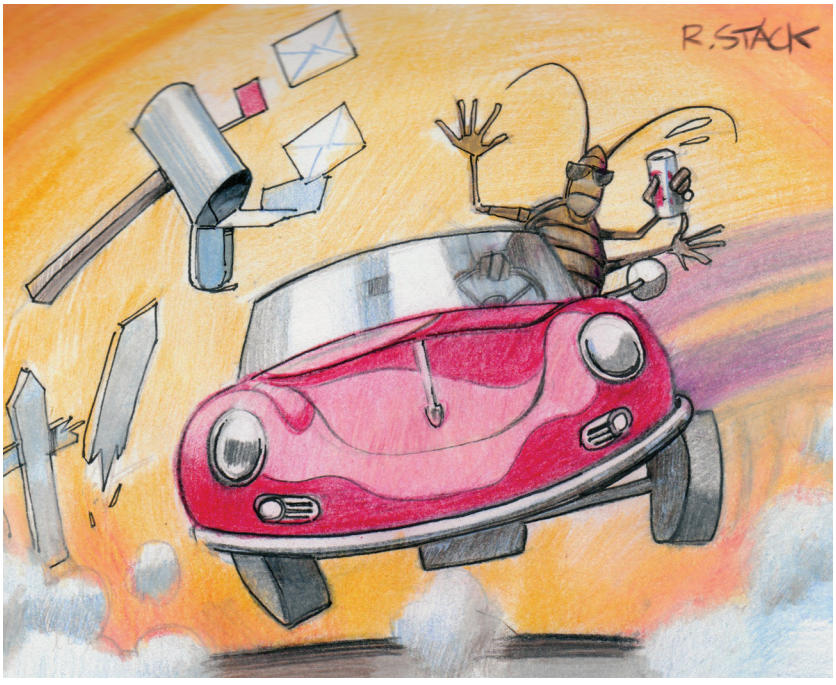
- decreased coordination, which slowly recovered after the policy was removed (although some teams recovered rather quickly).

According to Defeng Guo of Shanghai GiantStone Technology, "The results helped us reevaluate the pros and cons of using English as the mandatory lingua franca in our outsourcing development practices." You can access this paper at http://goo.gl/kbTXY9.

Crash-Inducing Commits

"An Empirical Study of Crash-Inducing Commits in Mozilla Firefox," by Le An and Foutse Khomh, shows that predictive models using machine-learning algorithms can identify crash-prone code at commit time, rather than after release.[3] Waiting until after release to prioritize crashes requires much data and can significantly delay fixing key functionality. These delays harm the user experience and increase defect repair effort because developers must recall important details about the underlying code.

An and Khomh's approach builds the predictive models using data from previous commits, including log metrics, code complexity metrics, social-network-analysis metrics, change type metrics, and whether the commit led to a crash. For projects that store commit history and crash reports in repositories, automated scripts can retrieve the data. The Python scripts An and Khomh used for Mozilla Firefox are at https://github.com/swatlab/crash-inducing. In this study, models created using the random-forests machine-learning approach achieved a median precision of 61.4 percent and a median recall of 76.5 percent for identifying crash-prone commits.



An and Khomh also identified factors involved in crash-inducing commits, including whether the commits

- were performed by less experienced developers,
- added or deleted large amounts of code,
- touched a large number of files, or
- contained larger commit messages.

Such information can help developers identify crash-prone code early. You can access this paper at http://goo.gl/cSs0D5.

UML Use

"On the Use of UML Documentation in Software Maintenance: Results from a Survey in Industry," by Ana Fernández-Sáez and her colleagues, presents the findings of a survey of 178 professionals from 12 countries.[4] Two important results were that

- 59 percent of the respondents used UML (and other graphical notations) for software maintenance and
- professionals used UML because they could obtain a better understanding of the system more quickly than without UML documentation. This better understanding improved defect detection.

The most popular UML tool was Visio, followed by Enterprise Architect and StarUML. This study also revealed factors that seemed to increase the use of UML diagrams: a higher level of education, working in larger teams, and maintaining larger systems. The survey results provide some insight into how companies might invest resources to ease maintenance tasks. You can access this paper at http://goo.gl/XX0Klu.

Simulating Legal Policies

"A Model-Based Framework for Probabilistic Simulation of Legal

Policies," by Ghanem Soltana and his colleagues, introduces a UML-based framework for legal-policy simulation.[5] Such simulation is an important decision-support tool to analyze how changes in the law affect measures of interest—for example, revenue. These simulations are important risk assessment tools for governmental agencies that create or modify laws and for companies impacted by those laws.

Legal-policy simulation currently combines spreadsheets and software code. Such a direct implementation poses a validation challenge. In particular, legal experts often lack the software background to review complex spreadsheets and code. So, they have no reliable means to check the simulations' correctness against the requirements envisaged by the law. Also, representative simulation data might be unavailable, necessitating a data generator.

Soltana and his colleagues' approach employs two UML profiles. The first profile enables the specification of executable legal policies through customized activity diagrams. The second profile captures the simulated population's probabilistic characteristics. This profile, which is applied to a class diagram specifying the core data concepts in the simulation population, supports the generation of synthetic simulation data.

The simulation framework lets nontechnical users test various legal-evolution scenarios in domains such as taxation and social security. Collaborating with legal experts, Soltana and his colleagues validated selected policy models from Luxembourg's tax law and demonstrated the simulation framework over the models. Thierry Prommenschenkel, a senior expert from Luxembourg's Inland Revenue Administration who was involved in the model validation, says, "With [the research team's] help, we have been able to put together the arguments and reasoning that tax practitioners deal with on a daily basis into a more consistent and understandable frame."

The project leverages the widespread use of models in the development of e-government applications. "For several years already, model-based methods have been part of our standard system development and maintenance practices, particularly business process modeling. Legal-policy simulation is a highly valuable addition to our existing model-based tools and will assist us in better predicting the impact of changes to laws and regulations," says Ludwig Balmer, who heads the Organization and Business Process Management Office at Luxembourg's Center for Information Technology. You can access this paper at http://goo.gl/tUwFcP.

References

1. X. Tang, S. Wang, and K. Mao, "Will This Bug-Fixing Change Break Regression Testing?," *Proc. 2015 ACM/IEEE Int'l Symp. Empirical Software Eng. and Measurement* (ESEM 15), 2015.
2. Y. Wang, "Language Matters," *Proc. 2015 ACM/IEEE Int'l Symp. Empirical Software Eng. and Measurement* (ESEM 15), 2015.
3. L. An and F. Khomh, "An Empirical Study of Crash-Inducing Commits in Mozilla Firefox," *Proc. 11th Int'l Conf. Predictive Models and Data Analytics in Software Eng.* (PROMISE 15), 2015, article 5.
4. A. Fernández-Sáez et al., "On the Use of UML Documentation in Software Maintenance: Results from a Survey in Industry," *Proc. 2015 ACM/IEEE Int'l Conf. Model Driven Eng. Languages and Systems* (MODELS 15), 2015, pp. 292–231.
5. G. Soltana et al., "A Model-Based Framework for Probabilistic Simulation of Legal Policies," *Proc. 2015 ACM/IEEE Int'l Conf. Model Driven Eng. Languages and Systems* (MODELS 15), 2015, pp. 70–79.

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

JORDI CABOT is an ICREA (Catalan Institution for Research and Advanced Studies) research professor at the Interdisciplinary Internet Institute. Contact him at jordi.cabot@icrea.cat.

LEANDRO L. MINKU is a lecturer (assistant professor) in the University of Leicester's Department of Computer Science. Contact him at leandro.minku@leicester.ac.uk.

MARCO TORCHIANO is an associate professor in Politecnico di Torino's Department of Control and Computer Engineering. Contact him at marco.torchiano@polito.it.