Test Prioritization and Localization at Ericsson

Overview of the Company-Specific Problem

A cellular base station connects cell phones to a voice and data network via LTE (4G) or 3G standards. The software that runs on these base stations contains not only complex signalling logic with stringent real-time constraints, but also must be highly reliable, providing safety critical services, such as 911 calling. At Ericsson Ottawa, testing base station software is time consuming and involves expensive specialized hardware. For example, testers may need to simulate cellular devices, such as when a base station is overwhelmed by requests from cell users at a music concert. In order to maximize the return on investment of Ericsson's testing efforts, we plan to use historical artifacts including past code changes, bugs, peer reviews, and test runs to create statistical models to (1) prioritize tests and (2) help developers categorize and locate the cause of a failure. The outcome of this work will advance the state-of-the-art in test prioritization and localization and provide Ericsson with a more efficient test infrastructure.

Background, Research Approach, and Data

The proposed work intersects two areas of software engineering: test prioritization and empirical software engineering. There are three streams of regression testing research [13]. The first, *minimization*, involves eliminating tests that are redundant or of low value. In the literature, the problem has been reduced to one of code coverage, for example, tests become redundant as the system evolves and more than one test covers the same control flow. As a result, much of the work in this area is algorithmic, such as transforming it into a spanning set problem [6], using divide-and-conquer strategies [1], and greedy algorithms [12]. The second, *selection*, uses the same static analysis techniques such as coverage [11] and slicing [3], but selects tests that cover source files that are at higher risk because they have been changed recently [8]. The third, *prioritization*, orders tests such that expensive, low-value, or long running tests are run after tests that find faults early. We focus on test prioritization because, at Ericsson, once a test fails an engineer must intervene to discover the fault, so an ordering that makes the test run fail early is more cost effective. While early prioritization techniques continued to use coverage measures to gauge priority, more recent approaches incorporate the faults found in past test runs [4] and change relationships among files [9] to identify high value tests.

We combine test prioritization techniques with empirical software engineering. Empirical researchers use historical data to create models to help developers, for example, identify bugs and risky changes [2], locate fault introducing changes [5], and understand the faults found during code review [7]. Relatively little work has combined test archives with information from other artifacts, such as bug reports and code reviews. One of the first studies to look at historical artifacts was done by a colleague at Concordia, Dr. Shihab, during his PhD. In this work, he created a statistical model from the source code history to help developers decide where to write tests for a legacy system at Blackberry [10].

A major factor limiting research into historically based test prioritization is information about past runs and links to other development artifacts. Ericsson collects massive histories of logs, test runs, and development artifacts, in part to provide for reproducibility of software and test lineups and auditing. Much of the current analysis is done manually where developers must sift through this information to locate the root cause of a failure. We plan to automate this process to help in prioritization test and localizing faults.

Research Competence

My research team has extensive training in mining and linking software development artifacts. We have worked extensively with semi-structured datasets that are similar to those at Ericsson and have been able to provide useful statistical models. For example, we have mined millions of code reviews, bug reports, tests results, and StackOverflow discussions. Rigby currently teaches these mining, linking, and analytics

techniques in a graduate course. Rigby has mined test data from Google Chrome in the past. While the Ericsson data will be richer, the techniques remain similar. The novel contribution lies in advancing our knowledge of test prioritization and localization and in improving the efficiency of quality assurance at Ericsson Ottawa.

Research Project and Technology Transfer

The proposal consists of three milestones, which will run for approximately 2 months each. First, we will investigate, clean, and link the available data. Second we will create a test prioritization model (RQ1). Third, we will create a fault localization model (RQ2).

Investigate Process and Data: A masters student will be onsite at Ericsson Ottawa. His goal will be to work with Griffiths's team to understand the quality assurance processes and to clean and link data. Ericssons in-kind contribution will be substantial as they will work with the masters student and help in interpreting the findings. I will visit Ericsson twice a month to help the student. A PhD student, Rahman, who researches release engineering, will also be investigating test prioritization and the impact on release effectiveness and help generalize the masters student findings to other software projects.

RQ1. Test prioritization model: Not all tests are equally important, but all tests must be run. With Ericsson's hard quality requirements, all tests must be run before a change can be released. However, some tests can span days, while others take seconds. Further, some tests have not failed in months, while some are new or fail regularly. Since a failure will stop the build process and require an engineer to intervene, a clear research direction will be to prioritize based on past failures and test running time [4, 14]. A simple ordering that makes low cost, high value tests fail early will make the process more efficient. Adding to these measures, we will develop change risk and social measures, such as risk analysis based on histories of defects across software areas and development teams. The research outcome of this stage will be a better understanding of the archival measures that provide good test prioritization.

RQ2. Fault categorization and localization model. Since fault localization is a wide research area including debugging techniques and statistical bug models, in this proposal, we focus on the narrow question – Did that test fail because of a problem in the development environment or product? This problem is unusually difficult at Ericsson because the environment is exceedingly complex. For example, test equipment is so expensive that developers in Ottawa can request their tests be run on specialized equipment that is only available in Sweden, and vice versa. Once a test fails, a developer must determine the root cause of the failure and whether it is an environment or product problem. Unfortunately, much of the other information in the report is added sporadically. We will investigate the reliability of existing fields in the reports and supplement this with freeform data extracted using existing NLP techniques. Our goal is to train a classifier based on past manual categorizations to help developers differentiate these failure types. In the future, we plan to refine this coarsegrained categorization to a finegrained localization at the file or method level by linking issue reports to other software artifacts.

Benefits to Canada

In 2009, Ericsson acquired the majority of Nortel's North American wireless access business, including its market share and became the world leader in this market. This human expertise remains in Canada where Ericsson employs about 1000 R&D staff in Ottawa. Our contribution will be to make these individuals more productive by improving the efficiency of the test process. If this NSERC Engage collaboration proves to be beneficial, we will extend the project through MITACS. We hope that this contribution leads to a long term relationship with Ericsson Ottawa where we can attack their practical problems and abstract them to the general advancement of knowledge in software engineering.

References

[1] T. Y. Chen and M. F. Lau. Dividing strategies for the optimization of a test suite. *Information Processing Letters*, 60(3):135 – 141, 1996.

[2] M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 31–41, May 2010.

[3] D. Jeffrey and N. Gupta. Test case prioritization using relevant slices. In *Proceedings of the 30th Annual International Computer Software and Applications Conference - Volume 01*, COMPSAC '06, pages 411–420, Washington, DC, USA, 2006. IEEE Computer Society.

[4] J.-M. Kim and A. Porter. A history-based test prioritization technique for regression testing in resource constrained environments. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 119–129, New York, NY, USA, 2002. ACM.

[5] S. Kim, T. Zimmermann, K. Pan, and E. J. J. Whitehead. Automatic identification of bug-introducing changes. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, ASE '06, pages 81–90, Washington, DC, USA, 2006. IEEE Computer Society.

[6] M. Marré and A. Bertolino. Using spanning sets for coverage testing. *IEEE Trans. Softw. Eng.*, 29(11):974–984, Nov. 2003.

[7] P. C. Rigby, D. M. German, L. Cowen, and M.-A. Storey. Peer review on open-source software projects: Parameters, statistical models, and theory. *ACM Trans. Softw. Eng. Methodol.*, 23(4):35:1–35:33, Sept. 2014.

[8] G. Rothermel and M. J. Harrold. A framework for evaluating regression test selection techniques. In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 201–210, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[9] M. Sherriff, M. Lake, and L. Williams. Prioritization of regression tests using singular value decomposition with empirical change records. In *Proceedings of the The 18th IEEE International Symposium on Software Reliability*, ISSRE '07, pages 81–90, Washington, DC, USA, 2007. IEEE Computer Society.

[10] E. Shihab, Z. M. Jiang, B. Adams, A. E. Hassan, and R. Bowerman. Prioritizing the creation of unit tests in legacy software systems. *Software: Practice and Experience*, 41(10):1027–1048, 2011.

[11] A.-B. Taha, S. Thebaut, and S.-S. Liu. An approach to software fault localization and revalidation based on incremental data flow analysis. In *Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International*, pages 527–534, Sep 1989.

[12] S. Tallam and N. Gupta. A concept analysis inspired greedy algorithm for test suite minimization. *SIGSOFT Softw. Eng. Notes*, 31(1):35–42, Sept. 2005.

[13] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012.

[14] Y. Yu, J. A. Jones, and M. J. Harrold. An empirical study of the effects of test-suite reduction on fault localization. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 201–210, New York, NY, USA, 2008. ACM.