An Empirical Examination of Abstract Test Case Prioritization Techniques

Rubing Huang*, Weiwen Zong*, Dave Towey†, Yunan Zhou*, Jinfu Chen*
*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China
{rbhuang, vevanzong, zhouyn, jinfuchen}@ujs.edu.cn
†School of Computer Science, The University of Nottingham Ningbo China, Ningbo, 315100, China
dave.towey@nottingham.edu.cn

***Abstract*—Abstract test case prioritization (ATCP) aims at ordering abstract test case in order to increase the speed at which faults are detected, potentially increasing the fault detection rate. This paper empirically examines possible ATCP techniques, according to the following four categories: *non-information-guided prioritization* (NIGP); *interaction coverage based prioritization* (ICBP); *input-model mutation based prioritization* (IMBP); and *similarity based prioritization* (SBP). We found that the ICBP category has better testing effectiveness than others, according to fault detection rates. Surprisingly, we found that NIGP can achieve similar performance to IMBP, and that SBP can sometimes achieve even better rates of fault detection than some ICBP techniques.**

***Index Terms*—Software testing, abstract test case prioritization, empirical examination**

I. Introduction

An *abstract test case* (ACT) [1] (also called a *model input* [2]) is an important type of test case, which can be derived from a model of the test object [3]. In combinatorial testing [4], for example, a test object is influenced by different factors (or parameters) with each factor containing a finite number of levels (or values), and ATCs can be obtained by choosing a level for each factor. ATCs have been widely used in many testing methods, such as category-partition testing [3], and model based testing [5]. *Abstract test case prioritization* (ATCP) has also been extensively studied in recent years, especially in the field of combinatorial testing [6–8] and software product line testing [9, 10].

This paper empirically examines possible ATCP techniques, which can be divided into the following four categories: *non-information-guided prioritization* (NIGP); *interaction coverage based prioritization* (ICBP); *input-model mutation based prioritization* (IMBP); and *similarity based prioritization* (SBP). Experiments were carried out to examine the testing effectiveness of each ATCP technique, the results of which are presented here.

II. ATCP Techniques Investigated

Based on the literature, we divide ATC prioritization techniques (ATCP) into four categories, according to the information used in each technique. Figure 1 shows the detailed information.

A. Non-Information-Guided Prioritization (NIGP)

The NIGP strategies can be used for all test cases (not only abstract test cases) because no additional information is required to guide the prioritization process.

- *Test-case-generation prioritization (TCGP):* prioritizes ATCs using the order in which the test case were generated.
- *Reverse test-case-generation prioritization (RTCGP):* prioritizes ATCs by reversing the generation order.
- *Random test case prioritization (RTCP):* prioritizes ATCs in a random manner, according to uniform distribution.

B. Interaction Coverage Based Prioritization (ICBP)

The ICBP strategy uses interaction coverage information to guide the ATCP process. Based on different levels of interaction coverage, three ATCP techniques were examined: *fixed-strength ICBP* (FICBP), *incremental-strength ICBP* (IICBP), and *aggregate-strength ICBP* (AICBP).

- *Fixed-strength ICBP (FICBP):* FICBP repeatedly chooses the next test case from candidates such that it covers the largest number of λ-wise factor-level combinations that have not yet been covered by previously selected ATCs. FICBP requires that the *prioritization strength*, an integer λ, be assigned in advance. Generally speaking, the prioritization strength is usually assigned a value between one and six [8, 11–14]. Additionally, we have recently proposed a new FICBP technique using repeated base-choice coverage (called FICBPR) [15]. FICBPR uses a similar mechanism to FICBP, but assigns the prioritization strength λ of 1, and forgets previously selected

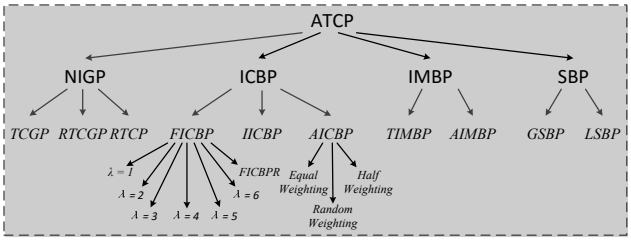

Fig. 1. ATCP Categories Investigated.

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.105

test cases when 1-wise factor-level combinations are fully covered.

- *Incremental-strength ICBP (IICBP):* IICBP [13, 14] is mainly used to prioritize ATCs for combinatorial test cases [4]. IICBP first uses the FICBP algorithm at a low prioritization strength λ_1 to prioritize the candidates. When all λ_1-wise factor-level combinations have been covered by selected test cases, IICBP increases the strength to λ_2 ($\lambda_2 > \lambda_1$), and then uses the FICBP algorithm with the new strength to guide prioritization of remaining ATCs. To make IICBP more normalized, λ_1 is generally set to 1, and λ_2 is set to $\lambda_1 + 1$ [14].
- *Aggregate-strength ICBP (AICBP):* AICBP [7] uses aggregated interaction coverage at different prioritization strength λ values, with λ ranging from 1 to the generation strength τ in combinatorial testing [4]. Since τ is determined for test suite construction, earlier AICBP versions may not be applicable to ATC sets. We used a normalized version of AICBP that only considers prioritization strength $\lambda = 1, 2, 3$, and can thus be used to prioritize any ATCs. The AICBP algorithm is similar to FICBP, except that AICBP uses aggregate-strength (1, 2, and 3) interaction coverage rather than a fixed-strength interaction coverage. Following previous studies [7], we adopted three weighting assignments: *equal weighting*, *random weighting*, and *half weighting*.

C. Input-model Mutation Based Prioritization (IMBP)

IMBP [2] first needs to mutate the constraints of the test object's input model, and then orders ATCs according to the mutant detection capability of each test case. The mutant operators used for the constraints follow previous studies [16]. Two IMBP techniques were examined: *total IMBP* (TIMBP) and *additional IMBP* (AIMBP) [2].

- *Total IMBP (TIMBP):* TIMBP follows previous '*total*' TCP strategies [17, 18], by prioritizing ATCs so that each selected candidate kills the maximum (total) number of model mutants.
- *Additional IMBP (AIMBP):* Similar to TIMBP, AIMBP follows previous '*additional*' TCP strategies [17, 18], by prioritizing ATCs so that each test case kills the maximum number of model mutants that have not yet been killed by already selected test cases.

D. Similarity Based Prioritization (SBP)

SBP [10] uses the Jaccard similarity between the candidates to order the test cases, with each ATC being considered as a set of factor levels. In particular, SBP chooses each next test case such that it has the least similarity to previously selected test cases. Two SBP techniques have been proposed by Henard et al. [10]: *global SBP* (GSBP) and *local SBP* (LSBP).

- *Global SBP (GSBP):* GSBP first chooses two candidates with the minimum similarity as the next two test cases, then selecting the one with minimum similarity to all previously selected test cases. For each candidate, GSBP sums the individual similarity values for the already selected ATCs, and then the candidate with the minimum sum value is chosen as the next element.
- *Local SBP (LSBP):* LSBP iterates over the candidate set of ATCs, identifying the two sharing the minimum similarity. The above process is repeated until all candidates are chosen.

III. Results

We used five open-source programs (obtained from the GNU FTP server [19]) written in the C language: flex, grep, sed, gzip, and make. These programs have been widely used in previous prioritization research [2, 8, 17, 20–23]. The ATCs are available from the Software Infrastructure Repository (SIR) [24]. Additionally, the well-known evaluation metric *average percentage of faults detected* (APFD) [17] was used to measure the rate of fault detection.

Based on mutation analysis, we have the following findings:

1) From the perspective of ATCP categories, ICBP has better testing effectiveness than other categories, in terms of fault detection capability. Somewhat surprisingly (because NIGP does not use any additional information), we found that NIGP can achieve similar performance to IMBP; and that SBP can obtain very good performance, sometimes even better than some ICBP techniques.
2) Among the ICBP techniques, we found that higher-strength FICBP techniques and IICBP have the best testing effectiveness, followed by AICBP and lower-strength FICBP techniques.
3) Among the IMBP techniques, we found that both TIMBP and AIMBP techniques have different performances for different object programs.
4) Among the SBP techniques, we found that GSBP has similar fault detection rates to LSBP.

IV. Conclusions

This study empirically examined 16 possible abstract test case prioritization (ATCP) techniques, along four categories, and presented the results.

Due to the page limitation, we only presented the comparison of fault detection rates among ATCP techniques, giving summary results rather than the empirical data. A larger scale empirical study is under way to investigate testing effectiveness and efficiency using all experimental data, and more evaluation metrics.

V. Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant No. 61502205 and 61202110, the Postdoctoral Science Foundation of China under grant No. 2015M581739 and 2015M571687, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under grant No. 15KJB520007, and the Senior Personnel Scientific Research Foundation of Jiangsu University under grant No. 14JDG039. This work is also supported by the Young Backbone Teacher Cultivation Project of Jiangsu University.

REFERENCES

[1] M. Grindal, B. Lindström, J. Offutt, and S. F. Andler, "An evaluation of combination strategies for test case selection," *Empirical Software Engineering*, vol. 11, no. 4, pp. 583–611, 2006.

[2] C. Henard, M. Papadakis, M. Harman, Y. Jia, and Y. L. Traon, "Comparing white-box and black-box test prioritization," in *Proceedings of the 38th Interaction Conference on Software Engineering (ICSE'16)*, 2016, pp. 523–534.

[3] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Communications of the ACM*, vol. 31, no. 6, pp. 676–686, 1988.

[4] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computer Survey*, vol. 43, no. 2, pp. 11:1–11:29, 2011.

[5] M. Utting and B. Legeard, "Practical model-based testing - a tools approach." *International Journal On Advances in Software*, vol. 2, no. 1, pp. 1–419, 2007.

[6] R. C. Bryce and C. J. Colbourn, "Prioritized interaction testing for pairwise coverage with seeding and constraints," *Information and Software Technology*, vol. 48, no. 10, pp. 960–970, 2006.

[7] R. Huang, J. Chen, D. Towey, A. Chan, and Y. Lu, "Aggregate-strength interaction test suite prioritization," *Journal of Systems and Software*, vol. 99, pp. 36–51, 2015.

[8] J. Petke, M. B. Cohen, M. Harman, and S. Yoo, "Practical combinatorial interaction testing: Empirical findings on efficiency and early fault detection," *IEEE Transactions on Software Engineering*, vol. 41, no. 9, pp. 901–924, 2015.

[9] M. Al-Hajjaji, T. Thüm, J. Meinicke, M. Lochau, and G. Saake, "Similarity-based prioritization in software product-line testing," in *Proceedings of 18th International Software Product Line Conference (SPLC'14)*, 2014, pp. 197–206.

[10] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y. L. Traon, "Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines," *IEEE Transactions on Software Engineering*, vol. 40, no. 7, pp. 650–670, 2014.

[11] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *IEEE Transactions on Software Engineering*, vol. 37, no. 1, pp. 48–64, 2011.

[12] R. C. Bryce and A. M. Memon, "Test suite prioritization by interaction coverage," in *Proceedings of the Workshop on Domain Specific Approaches to Software Test Automation (DoSTA'07)*, 2007, pp. 1–7.

[13] R. Huang, J. Chen, T. Zhang, R. Wang, and Y. Lu, "Prioritizing variable-strength covering array," in *Proceedings of the IEEE 37th Annual Computer Software and Applications Conference (COMPSAC'13)*, 2013, pp. 502–601.

[14] R. Huang, X. Xie, D. Towey, T. Y. Chen, Y. Lu, and J. Chen, "Prioritization of combinatorial test cases by incremental interaction coverage," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 10, pp. 1427–1457, 2013.

[15] R. Huang, W. Zong, J. Chen, D. Towey, Y. Zhou, and D. Chen, "Prioritizing interaction test suite using repeated base choice coverage," in *Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC'16)*, 2016, pp. 174–184.

[16] M. Papadakis, C. Henard, and Y. L. Traon, "Sampling program inputs with mutation analyais: Going beyond combinatorial interaction testing," in *Proceedings of the 7th International Conference on Software Testing, Verification and Validation (ICST'14)*, 2014, pp. 1–10.

[17] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929–948, 2001.

[18] L. Zhang, D. Hao, L. Zhang, G. Rothermel, and H. Mei, "Bridging the gap between the total and additional test-case prioritization strategies," in *Proceedings of the 35th Interaction Conference on Software Engineering (ICSE'13)*, 2013, pp. 192–201.

[19] GNU FTP Server. http://ftp.gnu.org/.

[20] X. Qu, M. B. Cohen, and K. M. Woolf, "Combinatorial interaction regression testing: A study of test case generation and prioritization," in *Proceedings of the 23rd International Conference on Software Maintenance (ICSM'07)*, 2007, pp. 255–264.

[21] B. Jiang, Z. Zhang, W. K. Chan, and T. H. Tse, "Adaptive random test case prioritization," in *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE'09)*, 2009, pp. 233–244.

[22] X. Qu, M. B. Cohen, and K. M. Woolf, "A study in prioritization for higher strength combinatorial testing," in *Proceedings of the 2nd International Workshop on Combinatorial Testing, (IWCT'13)*, 2013, pp. 285–294.

[23] J. Petke, M. B. Cohen, M. Harman, and S. Yoo, "Efficiency and early fault detection with lower and higher strength combinatorial interaction testing," in *Proceedings of the 12th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'13)*, 2013, pp. 26–36.

[24] H. Do, S. G. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Software Engineering*, vol. 10, no. 4, pp. 405–435, 2005.