Software development optimisation theory defined with graphs

Molan Gregor[1] and Molan Martin[2]

[1]Comtrade Digitral Services, Comtrade Research Group, http://comtrade.ai
[2]Faculty of Mathematics and Physics, Ljubljana, Slovenia

The aim of the software development process is to produce the best possible product with the given resources (money, time). As a part of the development process, quality assurance must also be rationalized. To do so, an abstract space is defined (software testing space), where software product is presented using graph theory. Test graph presents software product with all its functionalities. Test cases in a test graph are connections between vertices and vertices represent unit tests. Test suite and test phase are defined as subgraphs of test graph. The weights in test graph represent the cost and value of implementation for functionality. The first optimization algorithm (A_1), designed as the first step in the optimization of the software testing process, eliminates duplicated test cases. The second algorithm (A_2) alters the quantity of test cases for a given test phase. It is the method of drastically reducing the testing cost while jeopardizing the quality of the product. The third algorithm is a construction of an Optimal Test Phase (OTP), it is A_3 - OTP Construction. This optimization means that a maximum quality, given the resources, is reached. Depending on the circumstances algorithms A_1 and A_2, and A_1 and A_3 can be used together.

References

[1] M. J. Harrold, Testing: a roadmap, in: ICSE 00: Proceedings of the Conference on The Future of Software Engineering, New York, USA, ACM, 2000, 61–72.

[2] S. A. Sarcia, G. Cantone and V. R. Basili, Adopting curvilinear component analysis to improve software cost estimation accuracy model, application strategy, and an experimental verification, Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, 2008.

[3] E. M. Clarke and E. A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, Logics of Programs (1982), 52–71.

[4] J. P. Queille and J. Sifakis, A temporal logic to deal with fairness in transition systems, 1982, Article ID 1382758, 217–225.

[5] G. Fraser and F. Wotawa, Property relevant software testing with model-checkers, SIGSOFT Softw. Eng. Notes **31**(6) (2006), 1–10.

[6] V. Okun and P. E. Black, Issues in software testing with model checkers, 2003.

[7] D. R. Kuhn and D. R. Wallace, Software fault interactions and implications for software testing, IEEE Trans. Softw. Eng. **30**(6) (2004), 418–421.

[8] R. C. Bryce, A. Rajan and M. P. E. Heimdahl, Interaction testing in model-based development: Effect on model-coverage, (2006), 259–268.

[9] S. Sampath, R. C. Bryce, G. Viswanath, V. Kandimalla and A. G. Koru, Prioritizing user-session-based test cases for web applications testing, in: ICST '08: Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation, Washington, DC, USA: IEEE Computer Society, 2008, 141–150.

[10] C. B. Rene and J. C. Charles, The density algorithm for pairwise interaction testing: Research articles, Softw. Test. Verif. Reliab. **17**(3) (2007), 159–182.

[11] R. C. Bryce and C. J. Colbourn, Prioritized interaction testing for pairwise coverage with seeding and contraints, Information and Software Technology Journal (IST, Elsevier) **48** (2006), 960–970.

[12] K. Rick, L. Yu and K. Raghu, Practical combinatorial testing: Beyond pairwise, IT Professional **10**(3) (2008), 19–23.

[13] R. Kuhn and R. Kacker, Automated combinatorial test methods, (2008), 22–26.

[14] Wikipedia-Software-Testing, Software testing - wikipedia, the free encyclopedia, 2018, [Online; accessed 13-April-2018].

[15] IEEE, Ieee standard for software test documentation, 1998.

[16] Jakobsson, Free software project management howto, SoberIT - Software Business and Engineering institute, Tech. Rep., 2003. [Online]. Available: `http://www.soberit.hut.fi/T-76.115/02-03/palautukset/groups/pmoc/de/vmodel.pdf`

[17] B. Marick, New models for test development, 1999.

[18] B. M. Hill, V-model testing: Process model configuration using svg, Benjamin Mako Hill, Tech. Rep. 2008. [Online]. Available: `http://mako.cc/projects/howto/`

[19] M. Molan and G. Molan, Estimations of actual availability, 3-6 September 2001.

[20] J. Łukasiewicz, O logice trjwartos'ciowej (in Polish), Ruch filozoficzny **5** (1920), 170–171, English translation: On three-valued logic, in: L. Borkowski (ed.), Selected works by Jan Lukasiewicz, North-Holland, Amsterdam, 1970, 87–88.

[21] Wikipedia-Hirschberg's-Algorithm, Hirschberg's algorithm - wikipedia, the free encyclopedia, 2014, [Online; accessed 13-April-2018]. [Online]. Available: `http://en.wikipedia.org/wiki/Hirschbergs\_algorithm`

[22] D. S. Hirschberg, A linear space algorithm for computing maximal common subsequences, Commun. ACM **18**(6) (1975), 341–343. [Online]. Available: `http://doi.acm.org/10.1145/360825.360861`

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, 3rd ed. The MIT Press, 2009.