Combining Testing and Runtime Verification

Christian Colombo

Department of Computer Science, University of Malta

Testing and runtime verification are intimately related: runtime verification enables testing of systems beyond their deployment by monitoring them under normal use while testing is not only concerned with monitoring the behaviour of systems but also generating test cases which are able sufficiently cover their behaviour. Given this link between testing and runtime verification, one is surprised to find that in the literature the two have not been well studied in each other's context. Below we outline three ways in which this can be done: one where testing can be used to support runtime verification, another where the two techniques can be used together in a single tool, and a third approach where runtime verification can be used to support testing.

1 A Testing Framework for Runtime Verification Tools

Like any piece of software, runtime verification tools which generate monitoring code from formal specification have to be adequately tested, particularly so because of its use to assure other software. State-of-the-art runtime verification tools such as Java-MOP [12] and tracematches [3] have been tested on the DaCapo benchmark [2]. However, the kind of properties which have been monitored are rather low level contrasting with our experience with industrial partners who seem more interested in checking for higher level properties (such as the ones presented in [5, 4]). Whilst we had the chance to test our tool Larva [6] on industrial case studies, such case studies are usually available for small periods of time and in limited ways due to privacy concerns. Relying solely on such case studies can be detrimental for the development of new systems which need substantial testing and analysis before being of any use.

For this reason, we aim to develop a testing framework which would provide a highly configurable mock transaction system to enable thorough validation of systems which interact with it. Although not a replacement of industrial case studies, this would enable better scientific evaluation of runtime verification systems.

2 Combining Testing and Runtime Verification Tools

While testing is still the prevailing approach to ensure software correctness, the use of runtime verification [1] as a form of post-deployment testing is on the rise. Such continuous testing ensures that if bugs occur, they don't go unnoticed. Apart from being complementary, testing and runtime verification have a lot in common: runtime verification of programs requires a formal specification of requirements against which the runs of the program can be verified [11]. Similarly, in model-based testing, checks are written such that on each (model-triggered) execution step, the system state is checked for correctness. Due to this similarity, applying both testing and runtime verification

techniques is frequently perceived as duplicating work. Attempts [9] have already been made to integrate the runtime verification tool LARVA [6] with QuickCheck [10]. We plan to continue on this work by integrating the LARVA tool with a Java model-based testing tool, ModelJUnit[1].

3 Using Runtime Verification for Model-Based Testing Feedback

To automate test case generation and ensure that the tests cover all salient aspects of a system, model-based testing [7, 8] enables the use of a model specification from which test cases are automatically generated. Although successful and growing in popularity, model-based testing is only effective in as much as the model is complete. Sequences of untested user interaction may lead to huge losses for the service-providerif any of these lead to a failure. Since coming up with a model for test case generation is largely a manual process [7, 8], it is virtually impossible to ensure the model is complete.

In this context, we propose to use runtime information to detect incompleteness in the test case generation model: by considering the execution paths the system takes at runtime, a monitor checks whether each path (or a sufficiently similar one) is in the test case generation model.

References

1. Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G.J., Rosu, G., Sokolsky, O., Tillmann, N. (eds.): RV, LNCS, vol. 6418 (2010)
2. Blackburn, S.M., Garner, R., Hoffmann, C., Khang, A.M., McKinley, K.S., Bentzur, R., Diwan, A., Feinberg, D., Frampton, D., Guyer, S.Z., Hirzel, M., Hosking, A., Jump, M., Lee, H., Moss, J.E.B., Phansalkar, A., Stefanović, D., VanDrunen, T., von Dincklage, D., Wiedermann, B.: The dacapo benchmarks: java benchmarking development and analysis. SIGPLAN Not. 41(10), 169–190 (2006)
3. Bodden, E., Hendren, L.J., Lam, P., Lhoták, O., Naeem, N.A.: Collaborative runtime verification with tracematches. J. Log. Comput. 20(3), 707–723 (2010)
4. Colombo, C., Pace, G.J., Abela, P.: Compensation-aware runtime monitoring. In: RV. LNCS, vol. 6418, pp. 214–228 (2010)
5. Colombo, C., Pace, G.J., Schneider, G.: Dynamic event-based runtime monitoring of real-time and contextual properties. In: FMICS. LNCS, vol. 5596, pp. 135–149 (2008)
6. Colombo, C., Pace, G.J., Schneider, G.: Larva — safer monitoring of real-time java programs (tool paper). In: SEFM. pp. 33–37 (2009)
7. Dias Neto, A.C., Subramanyan, R., Vieira, M., Travassos, G.H.: A survey on model-based testing approaches: a systematic review. In: Empirical assessment of software engineering languages and technologies. pp. 31–36. WEASELTech '07 (2007)
8. El-Far, I.K., Whittaker, J.A.: Model-Based Software Testing. John Wiley & Sons, Inc. (2002)
9. Falzon, K.: Combining Runtime Verification and Testing Techniques. Master's thesis (2011)
10. Hughes, J.: Quickcheck testing for fun and profit. In: Practical Aspects of Declarative Languages, LNCS, vol. 4354, pp. 1–32 (2007)
11. Leucker, M., Schallhart, C.: A brief account of runtime verification. JLAP 78(5), 293–303 (2009)
12. Meredith, P.O., Jin, D., Griffith, D., Chen, F., Roşu, G.: An overview of the MOP runtime verification framework. JSTTT (2011), to appear.

[1] http://www.cs.waikato.ac.nz/ marku/mbt/modeljunit/