Exploring Possible Approaches and Challenges for the Extraction of Runtime Monitors from Tests

Luke Chircop*, Christian Colombo, Mark Micallef, Adrian Francalanza, and Gordon J. Pace

University of Malta - Department of Computer Science

The software development industry constantly strives to produce software which is reliable and of high quality. This is in most cases achieved through the use of popular techniques such as unit and integration testing. Although effective, such techniques do not ensure that all bugs are discovered and fixed. Other techniques that may provide extra guarantees exist but in most cases are not mainstream.

A case in point is runtime verification [4], which at runtime, is able to ensure that the system's behaviour is correct. Unlike testing, which checks a small subset of all the possible inputs, runtime verification is able to verify all execution runs as they occur. Although promising, the use of such a technique in industry is not widespread. This is mainly due to the extra resources and effort required to integrate the technology to the development process and the time it takes to write the actual properties for monitoring [2, 1].

This research project aims to make runtime verification more appealing to the software industry by making it easier to integrate and use. We intend to fulfil this objective by introducing an automated process as shown in Figure 1, which is able to process readily available information (such as tests and developer knowledge) and automatically generate runtime monitors.

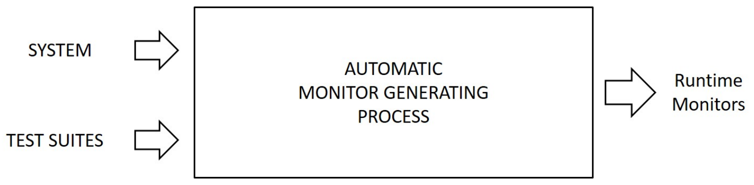

Fig. 1. High level representation of the automatic monitor generation process

Naturally, the proposed solution is not trivial and various challenges have been identified.

Designing the Automated Monitor Generation Process

A good selection of useful information regarding how the system should behave is essential for the automatic generation process to be successful. Various sources of such valuable information is in most cases available, ranging from — program specifications and test suites, to the actual source code and developer knowledge/experience.

Although the majority of software houses do design and document how the software to be developed should behave, in most cases they are not accurate, kept updated or complete. On the other-hand, test suites are generally present and up to date. Furthermore, such tests will by nature of their purpose contain valuable information with regards to how the system should behave and what inputs it can accept or reject. Therefore, we are hypothesising that tests could

* Project GOMTA financed by the Malta Council for Science & Technology through the National Research & Innovation Programme 2013

potentially serve as a great source of information to be exploited for the automatic generation of runtime verification monitors.

The idea of gathering tests and translating them into runtime monitors is not new. Pace and Falzon [3] presented a technique whereby QuickCheck automata used for model-based testing can be translated into runtime monitors. The downside to this approach however, is that like runtime verification, model-based testing is also not as widespread in industry.

With the aim of being as relevant as possible to the industry, we have therefore considered typical testing techniques such as unit, system, and integration tests. The translation from tests to runtime monitors is plausible due to the similarities that exist between the two: tests drive the system under test and check that the outcome is correct. Runtime monitors also check that the outcome is correct but lets the users drive the system. However, the similarities stop there. Tests are typically focused on checking very specific behaviour, whilst on the other hand runtime monitors are designed to handle all possible inputs. This means that direct translation from tests to runtime monitors will highly limit the effectiveness of the monitors generated.

Therefore, techniques capable of inferring conditions about how the system should behave have instead been considered. Such techniques typically represent conditions through the use of dynamic analysis and are formalised as invariants. Inferring invariants from a number of observed system executions is not new [5, 6]. Nonetheless, they are focused on generating invariants to automatically generate more test cases, debug and identify incorrect systems; not to be deployed with the system.

As a consequence, we have identified a number of cases whereby some of the inferred invariants are still too narrow, catering only for runs manifesting themselves during the testing phase. Therefore, apart from generating invariants, a filtering process is required whereby invariants which do not add value to the monitoring process are weeded out. In fact, a number of possible options dealing with invariant specificity are currently being explored.

Conclusion

The software industry is constantly trying to produce software that is of high quality and reliability. To this end, testing techniques are not enough to ensure that no more bugs exist. Runtime verification is seen as a technique that could provide the extra assurance but is barely used due to some limiting factors. In this abstract we have introduced an approach to automatically generate runtime monitors through the use of dynamic analysis in the hopes of making runtime verification more appealing for the software industry. Furthermore, we have identified a couple of limitations which are going to be tackled in the near future to make the process even more refined.