Flexible and Extensible Runtime Verification for Java

Runtime verification validates the correctness of a program's execution trace. Much work has been done on improving the expressiveness and efficiency of runtime verification. However, current approaches require static deployment of the verification logic and are often restricted to a limited set of events that can be captured and analyzed, hindering the adoption of runtime verification in production systems. A popular system for runtime verification in Java, JavaMOP (Monitor-Oriented Programming in Java), suffers from the aforementioned limitations due to its dependence on AspectJ, which supports neither dynamic weaving nor an extensible join-point model. In this article, we extend the JavaMOP framework with a dynamic deployment API and a new MOP specification translator, which targets the domain-specific aspect language DiSL instead of AspectJ; DiSL offers an open join-point model that allows for extensions. A case study on lambda expressions in Java8 demonstrates the extensibility of our approach. Moreover, in comparison with JavaMOP using load-time weaving, our implementation reduces runtime overhead by 32%, and heap memory usage by 13%, on average.

Author: Walter Binder (walter.binder@usi.ch)