정적 분석기의 동적 생존 전략

이찬구

훌륭한 정적 분석기가 살아남는 것이 아니라, 살아남은 정적 분석기가 훌륭한 것이다. 효율과 안전 사이에서 줄다리기하는 소프트웨어 개발 과정에서 정적 분석기 사용을 마냥 강요할 수는 없기 때문이다. 구글(Google), 메타(Meta), 애플(Apple)은 정적 분석기를 개발 과정에 자연스럽게 녹여내기 위해 노력했고, 그 결과 트리코더(Tricorder)와 인퍼(Infer) 같은 도구들은 현재까지 살아남을 수 있었다. 하지만 인공지능(artifical intelligence)과 데브옵스(devOps) 기술의 발전으로 인해 정적 분석기는 또 한 번의 도전에 직면하고 있다. 본 글에서는 정적 분석기가 그동안 선택해 온 전략을 살펴보고, 변화하는 개발 환경 속에서 앞으로 어떤 전략을 취해야 할지 논의한다.

육상의 최강자로 군림했던 공룡은 멸종했지만, 인간에게 유용한 존재였던 닭은 그 뒤를 이어 오늘날까지도 번성하고 있다. 정적 분석기의 운명도 이와 비슷하다. 실행범위(coverage)가 넓고 많은 버그를 찾아낼 수 있는 정적 분석기가 반드시 살아남는 것은 아니다. 중요한 것은 개발자들이 이를 사용할 의사가 있는가이다. 대규모 소프트웨어를 개발하는 구글, 메타, 애플 역시 정적 분석기의 성공 여부는 단순한 기술적 우수성이 아니라, 개발자들이 이를 얼마나 받아들이고 활용하는지에 달려 있다고 지적했다.

파인드버그(FindBugs)는 나쁜 정적 분석기가 개발자들에게 외면받는 이유를 여실히 보여주었다. 그리고 그 대가로 구글에서는 세 번의 정착 실패를 겪었다. 원인 중 하나는 치명적이지 않거나 즉각 수정할 수 없는(not actionable) 버그에 대한 정적 분석기의 경고들이었다. 이들은 효율을 중시하는 대부분 개발자에 의해 무시되어 버그 수정(debugging)이라는 목표를 달성하지 못하고 결과적으로 오탐(effective false positive)이 되었다. 또한, 기존의 작업흐름(workflow)과 지나치게 동떨어진 방식으로 버그 수정이 요구된다면 개발자들에게 추가적인 부담이 되며, 이 또한 정적 분석기의 사용을 꺼리게 만드는 원인이 되었다.

이러한 실패에서 교훈을 얻은 구글의 트리코더는 개발자 작업흐름에 자연스럽게 녹아들도록 코드 리뷰 시점과 컴파일 시점에 수정이 필요한 버그들을 제안하였고, 제안들은 버그의 중요도를 고려하여 각 시점에 배치되었다. 대상 소프트웨어의 크기와 용도는 달랐지만 메타의 Infer 역시 개발자들이 자연스럽게 정적 분석 결과를 받아들이게 했다. 인퍼는 버그를 실시간으로 분석, 수정 결과까지 제안하면서 개발자의 사용성을 극대화하고 부담을 최소화하였다. 애플이 제안한 엑스코드(Xcode)의 정적 분석기는 더욱 직관적이다. 버튼 한 번만 누르면 실행되며, 필요할 때만 개발자가 선택적으로 실행할 수 있어 불필요한 방해 없이 정적 분석을 활용할 수 있다.

그렇다면 이제 정적 분석기는 개발자들의 외면에서 안심해도 되는걸까? 그렇지 않다. 최근 인공지능 기반의 코드 자동 완성(auto completion)이나 지속적 통합 및 배포(continuous integration/continuous deployment)같은 데브옵스 기술 발전이 정적 분석기에 또 다른 위기를 가져오고 있다. 개발자는 점점 더 자동화된 환경에서 효율적으로 일하려 한다. 변화된 개발자의 태도와 작업흐름에 적응하기 위하여 정적 분석기는 다시 한번 진화해야 한다.

현대의 개발자들은 단순히 버그를 찾아주는 도구가 아니라, 실질적으로 문제 해결을 도와주고 생산성을 높이는 도구를 원한다. 따라서 정적 분석기는 단순한 버그 탐지에 머무르지 않고, 자동 수정(auto-fix) 기능을 정교하게 만들고, 인공지능 기반 코드 분석을 활용하여 문맥에 맞는 조언을 제공해야 한다. 물론 이를 가능하게 하기 위한 정적 분석에서 인공지능의 역할과 안전한 코드 생성은 필수적으로 연구돼야 할 과제일 것이다. 그뿐만 아니라, 자동화된 개발 흐름, 배포 흐름과 더욱 긴밀하게 결합하여, 개발자가 별도로 실행해야 하는 도구가 아니라 자연스럽게 작동하는 환경을 조성해야 한다. 이와 동시에 새로운 환경에서 개발자가 필수적으로 개입해야하는 때는 언제인지, 완전 자동화가 되어도 괜찮을 때는 언제인지 판단하는 것도 중요하다. 구글의 실패 경험을 답습하지 않기 위해서 말이다. 이런 모든 판단에 있어서도 인공지능이 활용될 수 있는지, 활용되어도 되는지는 아직 미지의 영역이다.

완벽한 정적 분석기는 존재하지 않는다. 개발 흐름이 변화하는 이상, 정적 분석기도 변화해야 한다는 의미다. 인공지능과 데브옵스 기술의 발달로 개발자의 역할이 빠르게 변화하는 가운데,

어떻게 더 효율적이고 안전한 도구로 자리 잡을 것인가라는 문제는 끊임없이 되물어야 한다. ’새로운 소프트웨어에서는 어떤 버그를 찾아야 하는가?’ ’새로운 개발 흐름에 녹아들기 위해서는 어떻게 해야하는가?’ ’인공지능이 활용되었을 때 안전한 코드 생성은 어떻게 추구할 수 있는가?’ 현재 얻은 답에 만족한 채 안주한다면 과거 구글이 겪었던 실패를 되풀이하는 것은 시간문제일 것이다.