정적(靜的)이며 정적(情的)인

김세훈

훌륭한 도구와 훌륭하지 않은 도구를 구분 짓는 것은 도구 자체의 성능뿐만이 아니다. 도구는 '사람에 의해 사용되는 것'이라는 본질은 도구의 사용성(usability)이 도구를 평가하는 중요한 성질임을 암시한다. 정적 분석 도구(static analysis tool)를 개발자 작업 흐름(workflow)에 통합하려는 구글(Google), 페이스북(Facebook) 및 애플(Apple)의 노력은 사용자가 접근하기 쉽고, 사용하기 편하고, 믿을 수 있는 방향으로 도구를 개선하는 데 초점을 맞추었다. 한 개인이 어떤 도구를 사용할 것인가 사용하지 않을 것인가에 대한 결정은 그 도구를 사용함으로써 얻는 손익을 비교하여 얻어진다. 심리나 습관을 포함한 인간에 대한 깊이 있는 이해를 정적 분석 도구에 접목하면, 도구의 성능적인 향상 없이도 프로그램 버그 탐지와 수정에 실질적인 도움이 된다.

정적(靜的) 분석 도구는 도구이고, 도구는 곧 사용하는 것이다. 사용되지 않는 도구는 존재 의의가 없다. 많은 프로그램 버그가 정적 분석 도구를 통해 조기에 발견되고 수정될 수 있음에도, 개발자들이 좀처럼 이러한 도구 사용을 거부하거나 무시하는 이유는 그 도구가 형편없기 때문이다. 그리고 이 '형편없는 것'에는 느린 분석 속도나 높은 오탐률(false positive rate) 등 도구 자체로서의 성능 결함뿐만 아니라, 사용자에 대한 고려 결여, 즉 인간에 대한 몰이해를 포함한다. 아무리 검날을 날카롭게 연마했다고 해도 손잡이가 사람의 손에 맞지 않으면 아무도 그 검을 쓰려하지 않을 것이다. 개발자가 정적 분석 도구를 사용하게 만들기 위해서는 정적 도구의 손잡이부터 뜯어고쳐야 한다.

사람은 감정적(感情的)이다. 번거로운 것은 피하려고 하고, 반드시 하지 않아도 될 것은 안 하고 넘겨 버린다. 거짓말쟁이의 말에 귀를 기울이지 않고, 실제로 눈에 보이는 것만 믿는다. 오래전 일은 잊어버리려 하고, 자신과 무관한 일에는 관여하고 싶어 하지 않는다. 구글의 팀 BugBot은 Java의 정적 분석 도구인 FindBugs를 개발자의 작업 흐름에 통합시키기 위해서 바로 위에서 언급한 인간적 요소들을 공략했다. 수차례의 시행착오 및 개발자 피드백을 거쳐 BugBot은 컴파일러와 코드 리뷰 도구에 정적 도구를 통합시키는 데 성공했다. 페이스북 또한 Infer 및 Zoncolan이라는 정적 분석 도구에 변경 시점 분석(diff-time analysis)을 도입하여 신속한 버그 보고를 통해 개발자의 작업 흐름이 방해받지 않도록 했다. 애플은 통합 개발 환경(integrated development environment)인 Xcode에 정적 분석 도구를 내장하여 도구에 대한 접근성 및 편의성을 크게 향상했다.

위 모든 노력이 정적 분석 도구 자체의 개선이 아니라 작업 흐름 상에서 도구를 사용할 시점이나 사용자 편의성 개선 등 외부적 요소에서 기인한다는 사실은 흥미롭다. 코파일럿(Copilot), ChatGPT 등 이미 여러 대형 언어 모델들이 자연어로부터 자동으로 코드를 생성해 내고 있지만, 상용구 코드(Boilerplate code) 수준을 넘어서면 아직은 개발자를 보조하는 수단에 불과하다. 결국 인간을 뛰어넘는 강인공지능이 출현하기 전까지는 프로그램은 인간의 감독하에 작성되고 수정되어야 하는 것이다. 프로그램 개발 과정에서 인간의 개입이 필수적이라면, 그 개입 과정에서 어떤 도구를 사용하는지가 최종 산출물의 질을 결정한다. 그리고 눈앞에 도구가 있음에도 사용하지 못하는 게으른 사람들을 위해서 해야 할 일은 도구를 더 휘황찬란하게 만드는 게 아니라, 도구를 직접 손에 쥐어 주는 것이다.

물론 이 모든 것은 도구 자체가 훌륭해야 한다는 전제가 우선된다. 도구의 분석 시간을 줄이는 방향으로 개선되어야 함은 물론, 오탐률 및 미탐률(false negative reate)을 줄이고 반대로 정탐률(true positive rate)을 높여야 한다. 가능하다면 버그를 보고하는 데 그치지 않고, 버그에 대한 다양한 수정 사항을 직접 제시하거나 관련된 참고 자료를 표시할 수 있어야 한다. 그리고 대다수 정적 분석은 분석이 적용되는 분야에 따라 전체적인 구현 방식과 세부 사항이 다르게 설정되어야 하므로, 해당 분야의 전문가와 정적 분석을 사용할 사용자에게서 적극적으로 피드백을 수용하는 것이 중요하다.

위 논의는 단순히 정적 분석에만 한정되는 이야기가 아니다. 유닛 테스트, 퍼징(fuzzing), 회귀 테스트(regression test) 등 다양한 동적 분석에 사용되는 도구나 제품 출시 후 사용자로부터 받는

버그 리포트에 대해서도 같은 논리를 적용할 수 있다. 단순히 더 좋은 분석 도구를 개발하는 데에만 사고를 제한하지 않고 개발 도구를 사용하는 사람의 입장에서 생각해본다면 도구는 개발자와 상호작용하며 창의적인 방식으로 도구의 유용성을 증대시킬 수 있을 것이다.