인간의 불완전함과 정적분석의 미래

강찬우

2024-03-03

정적분석 도구는 소프트웨어 개발자가 코드 내에 존재하는 오류를 빠르고 쉽게 찾아낼 수 있도록 돕는다. 미국의 거대 IT 기업인 구글과 페이스북은 각각 자사만의 정적분석 과정을 도입하여 성공적으로 개발자들이 정적분석 도구의 도움을 받고, 이를 해결하기 위한 큰 비용을 절약할 수 있도록 하였다. 그러나 이 정적분석 도구들은 아직 사람의 신뢰와 도움에 의존한다는 점에서 불완전함을 지니고 있다. 이 글에서는 코드의 양이 더욱 많아지고 복잡해질 미래에는, 정적분석 도구가 오류 여부의 판단과 코드의 수정을 오로지 인간에게 의존하는 것으로부터 탈피하여야 함을 역설한다.

미국에는 이른바 FAANG이라고 불리는, 거대 IT 기업이 존재한다. 이들은 하루에만 수만 줄의 코드를 작성하고, 수정한다. 이 과정에서 필연적으로 수많은 소프트웨어 오류가 생성된다. FAANG에서 각각 G와 F를 담당하는 IT 회사인 구글과 페이스북에서는 각각 Tricoder, Infer 등의 소프트웨어 정적분석 도구를 만들어 이러한 소프트웨어 오류를 찾기 위해 노력하였다. 또한 이들은 이 도구를 효과적으로 사용한 성공적인 이야기를 공유하기도 하였다.

이들의 이야기를 들어보면, 공통적인 부분을 많이 발견할 수 있는데, 나는 그중에서 특히 '분석 도구의 결과에 대한 소프트웨어 개발자의 신뢰와 분석 도구의 결과가 실질적인 코드 수정으로 연결되기까지의 시간이 매우 중요하다'라는 부분에 집중하고 싶다. 이들은 모두 소프트웨어 오류 해결에 있어서 사람의 역할이 중요함을 강조하는 것이기 때문이다.

아무리 코드 내 오류를 잘 찾아내는 정적분석 도구가 존재한다고 하더라도, 이를 인간이 신뢰하지 못하고, 활용하지 않는다면 그 도구는 성공했다고 볼 수 없다. 이는 구글에서 그들이 만든 정적분석 도구인 FindBugs의 사례에서도 알 수 있다. 그들은 실패를 인정하였고 그 경험으로부터 '성공한' 도구를 제작하였다. 그 도구인 Tricoder는 단순히 정적분석을 통해 잠재적인 오류를 알려주는 것을 넘어, 개발자들의 개발 환경과 동화되고 그들의 의견을 적극 반영하여 확장될 수 있는 하나의 시스템이었다.

페이스북에서의 사례도 구글의 사례와 마찬가지로 개발 환경에 동화되어, 그들이 수정한 코드가 모두 확정(commit)되기 이전에 분석 도구에 의해 검증되어야 한다. 구글에서는 단순한 정적분석 도구를 적용했던 것과 달리, 좀 더 복잡하게 동작하는 방식을 적용한 정도의 차이가 있다.

그러나 나는 두 사례가 정적분석 분야에 좋은 방향을 제시하였음을 인정하는 동시에, 이것으로는 아직 부족함을 주장하고 싶다. 앞서 언급한 바와 같이 이들은 모두 '사람'에 의존하고 있기 때문이다. 정적분석 도구는 소프트웨어 오류가 고쳐지기 위해 '사람의 신뢰'를 얻어야 하며, 그들이 '귀찮아하지 않도록' 코드를 작성한 즉시(혹은 적어도 매우 늦지 않은 시간 안에), 혹은 코드 검토 시간을 활용해서 수정을 제안해야 한다.

소프트웨어가 더욱 발전하여 코드의 양이 더욱 많아지고 복잡해진 미래를 상상해 보자, 그때는 지금의 두 사례와 같이, 사람에게 의존하는 정적분석은 또 다른 원인에서 기인한 실패를 맞이할 것이다. 그 원인은 역설적이게도 그 정적분석을 신뢰하고 검증하는 이들이 사람이라는 사실이다. 사람은 '불완전'하기 때문이다. 사람은 때때로 실수하며, 일할 수 있는 시간도 한정적이다.

해결 방법으로, 더 좋은 정적분석 도구를 만들어 미탐(false negative)과 오탐(false positive)의 비율을 줄이는 것을 생각해 볼 수 있다. 이와 같은 접근은 코드의 양이 더 많아지더라도 사람이 해야 할 일을 다시 줄일 수 있기 때문이다. 그러나 이와 같은 접근법 역시 한계가 있으며 '완전'하지 못하다.

대신에 이 과정에서 인간을 완전히 배제해야 한다. 정적분석 도구가 잠재적인 오류를 탐지하면, 이를 도구 스스로가 다시 평가하여 오탐의 여부를 판단해야 한다. 그 후 높은 확률로 오류라고 판단이 되면 즉시 스스로 코드를 수정해야 한다. 실제로 분석 도구가 이렇게 동작하게 된다면, 코드 내 오류의 탐지와 코드 수정 사이의 시간적 격차를 신경 쓰지 않아도 된다. 이는 곧 분석 도구가 충분한 시간을 들여 오류를 탐지할 수 있도록 할 것이며, 이는 더욱 고도화된 정적분석을 하여 다시 미탐와 오탐를 줄이는 결과로 나타날 것이다.

분석 도구가 코드의 오류에 대한 탐지 결과에 대해 스스로 오탐 여부를 판단할 때, 임곗값을 설정하여 그 이상이면 스스로 코드를 수정하고, 그 아래이면 기존과 같이 수정을 요청하는 식으로 구현하는 것도 생각해볼 수 있다. 그렇게 되면 기존의 방법도 복합적으로 활용할 수 있을 것으로 보인다. 또 이 과정에서 기계 학습을 사용하여 임곗값을 조절하면, 이 임곗값의 최적점을 찾도록 학습시킬 수 있을 것이다.

복잡하게 얽혀있는 코드의 늪에서, 정적분석 도구는 소프트웨어 개발자에게 많은 시간과 노력을 아끼는 데에 큰 도움이 된다. 그러나 머지않은 미래에 더욱 코드의 양이 많아지고 복잡해질 것을 대비해 사람의 간섭이 필요하지 않은 정적분석 도구를 만들어내는 것이 필요하게 될 것으로 예상한다.