# 소프트웨어 스멜 검출 도구의 낮은 사용률 개선을 위한 연구

주한새° Scott Uk-Jin Lee
한양대학교 바이오인공지능융합전공
{sparky, scottlee}@hanyang.ac.kr

## 1. 서론

소프트웨어 스멜은 일반적으로 소프트웨어 품질 속성에 부정적인 영향을 미치는 증상을 나타내는 징후나 지표이며, 소프트웨어 유지보수성에 악영향을 끼친다고 알려져 있다[1,2]. 따라서 스멜을 검출하는 것은 리팩토링, 즉 소프트웨어 진화 과정을 개선하는 결정을 내릴 때 중요하게 작용한다. 하지만 현재 소프트웨어 스멜 검출 도구를 사용하는 비율은 미약하다[3]. 본 논문에서는 현재 스멜 검출 도구 사용률이 낮은 원인을 분석하고 이를 개선하기 위한 방안을 제시한다.

## 2. 스멜 검출 도구의 낮은 사용률 분석

많은 개발자 및 연구자들이 스멜 검출 도구를 사용하기 보다 리팩토링 시에만 도구를 활용하고 있는데[3], 그 원인을 다음과 같이 분석하였다.

1) 스멜 검출 도구는 IDE 의 확장프로그램의 형태로 제공되는 경우가 많은데, 다수가 Eclipse 로만 제공되고 IntelliJ, VS 등의 다른 IDE 나 편집기에는 지원되지 않아 IDE 종속적인 개발자들에게 접근성이 떨어진다.
2) 많은 개발자들이 자신의 개발환경에 맞춰 프로그래밍 도구들을 자동화하지만, 대부분 GUI 로만 제공되어[3] 자동화에 어려움이 있다. 특히, 스멜 연구자들에게는 대규모 스멜 연구를 위해서 자동화가 필수적이다.
3) 소스코드나 프로그램을 제공하지 않거나 유지보수의 부재로 사용 불가능한 스멜 검출 기법 연구가 많다. 특히, 최근 활발히 연구되는 머신/딥 러닝 기반의 도구는 WEKA, TensorFlow 등으로 작성된 소스코드나 학습 데이터, 혹은 학습된 모델이 공개적으로 이용가능하지 않은 경우에 동일한 성능을 보장하기 어렵다[4].
4) 스멜 탐지 기법은 각 접근법 유형(메트릭, 논리/룰. 머신러닝, 시각화, 그래프 기반 등)에 따라 성능 지표(precision, recall 등)가 달라 성능 비교에 동질성이 결여되어 있다[3].
5) 현재 커뮤니티는 스멜 검출법이 높은 거짓 양성률(FP)을 보이고 있다고 믿고 있다[2].

## 3. 개선 방안

본 연구에서는 스멜 검출 도구들을 콘솔 TUI 기반의 독립 프로그램으로 Docker 나 Podman 등의 컨테이너 이미지화 하여 배포할 것을 제안한다. 이를 통해 사용자들이 어떤 개발환경을 갖추고 있어도 스멜 검출 도구를 통합하고, 쉽게 자동화 및 관리가 가능하다. 또한, 다양한 스멜 검출 도구들의 성능 지표를 한 번에 비교 분석하는 벤치마크 등을 만들기 쉽고, 연구에서 실험한 것과 동일한 환경에서 실행 가능하여 어느정도 동일한 성능을 보장할 수 있다. 이를 통해 더 많은 개발자 및 연구자들이 사용할 것이고, 스멜 검출 연구 커뮤니티의 발전으로 이어져 소프트웨어 스멜 검출법의 높은 FP 같은 문제점도 극복할 수 있다.

## 4. 결론 및 향후 연구 계획

본문에서는 소프트웨어 스멜 검출 도구가 사용률이 낮은 원인을 다음과 같이 분석하였다.

1) IDE 종속성으로 인한 낮은 접근성
2) 자동화 난이도가 높은 GUI 도구가 많음
3) 사용 불가능한 소스, 프로그램, 데이터
4) 서로 다른 성능 지표로 공정한 비교 어려움
5) 검출 도구의 FP 가 높다는 커뮤니티의 인식

위 문제점을 개선하기 위한 방안으로 소프트웨어 스멜 검출 도구를 TUI 독립 프로그램을 컨테이너 이미지화 하여 배포해야 한다. 이를 통해 전체적인 소프트웨어 업계의 유지 보수성 향상을 기대할 수 있다. 향후 연구는 현업 개발자들과 연구자들을 대상으로 제안한 개선안의 효용성 검증을 위한 경험적 연구로, 현재 계획 중에 있다.

## 5. 사사

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2020R1F1A1076208).

## 6. 참고 문헌

[1] Fowler M., Beck K., Brant J., Opdyke W., and Roberts D. 1999. Refactoring: Improving the Design of Existing Code (1st ed.). Addison-Wesley Professional, 1999
[2] T. Sharma and D. Spinellis, “A survey on software smells,” *J Syst Software*, vol. 138, pp. 158–173, 2018, doi: 10.1016/j.jss.2017.12.034.
[3] . K. Alkharabsheh, Y. Crespo, E. Manso, and J. A. Taboada, “Software Design Smell Detection: a systematic mapping study,” *Software Qual J*, vol. 27, no. 3, pp. 1069–1148, 2019
[4] A. Al-Shaaby, H. Aljamaan, and M. Alshayeb, “Bad Smell Detection Using Machine Learning Techniques: A Systematic Literature Review,” *Arab J Sci Eng*, vol. 45, no. 4, pp. 2341–2369, 2020