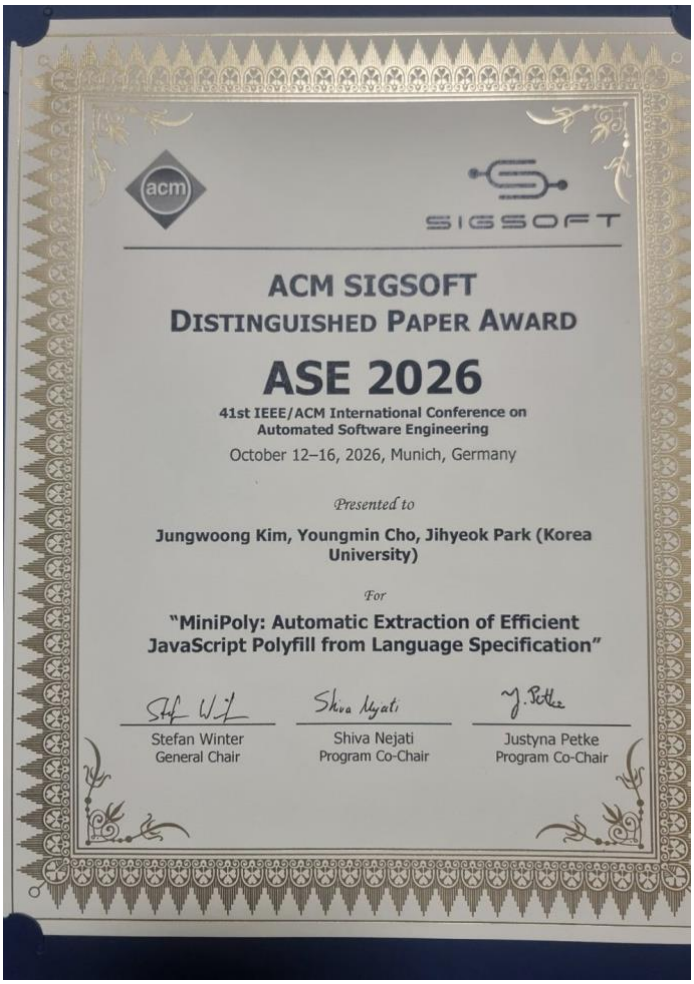
ASE 2026 Trip Report

Munich, Germany

October, 12-16, 2026

조영민



ACM SIGSOFT
DISTINGUISHED PAPER AWARD

ASE 2026

41st IEEE/ACM International Conference on Automated Software Engineering

October 12–16, 2026, Munich, Germany

Presented to

Jungwoong Kim, Youngmin Cho, Jihyeok Park (Korea University)

For

"MiniPoly: Automatic Extraction of Efficient JavaScript Polyfill from Language Specification"

Stefan Winter
General Chair

Shiva Nejati
Program Co-Chair

Justyna Petke
Program Co-Chair

Introduction

ASE(IEEE/ACM International Conference on Automated Software Engineering)는 프로그램 분석, 테스팅, 검증, 유지보수처럼 소프트웨어 개발을 자동화하는 기법과 도구를 다루는 소프트웨어 공학 분야의 대표 국제 학회다. 41 회째인 올해 ASE 는 10 월 12 일부터 16 일까지 독일 뮌헨에서 열렸고, 13~15 일 본 학회 앞뒤로 워크숍이 함께 진행됐다. 이번 학회 참석의 가장 직접적인 이유는 우리 논문 MiniPoly 의 발표와, 이 논문이 받게 된 ACM SIGSOFT Distinguished Paper Award 시상이었다. 그와 별개로 개인적으로는 AI 가 소프트웨어 개발 방식을 실제로 어떻게 바꾸고 있는지가 궁금했다. AI 로 특정 문제를 잘 풀었다는 사례보다는, 학계에서 이 변화를 큰 흐름에서 어떻게 보고 있는지 알고 싶었다.

이번 여름 삼성전자에서 인턴을 하며 개발 조직에 AI 를 도입하는 일을 가까이에서 보았는데, 그때 가장 답이 없던 질문이 "AI 를 쓰면 개발자의 성과는 무엇으로 측정해야 하는가", "효율이 실제로 얼마나 올랐는가"였다. 현업에서는 아직 감으로 이야기하는 부분이 많았기에, 학계는 이 질문을 어떻게 다루고 있는지 확인하고 싶었다.

함께 논문을 쓴 정웅이와, 학회에 갈 수 있도록 지원해주시고 논문을 끝까지 함께 다듬어주신 박지혁 지도교수님께 감사를 전한다.

Interesting Talks

Overall

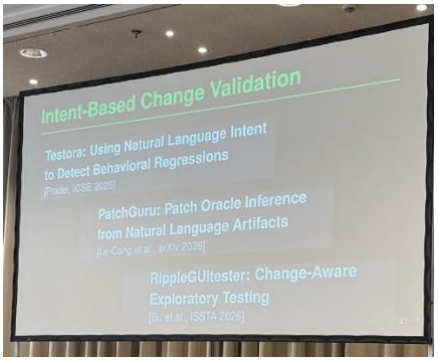
내 기대와는 조금 결이 달랐다. 발표의 상당수가 특정 문제를 LLM 으로 해결한 사례 중심이었고, 내가 보고 싶었던 "AI 가 개발 프로세스를 어떻게 바꾸고 있는가"라는 큰 그림을 다루는 발표는 생각보다 드물었다.

그래도 돌이켜보면 기억에 남는 발표가 몇 개 있었다.

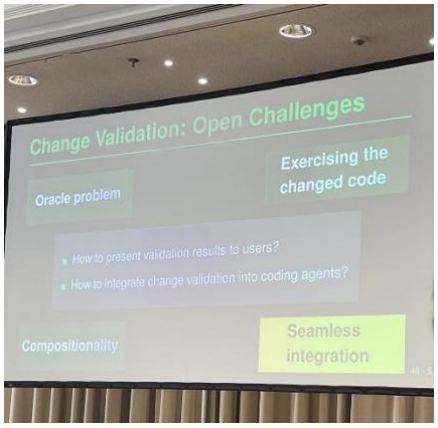
Trust the Delta: Rethinking Software Validation in the AI Era

첫날 키노트로 CISPA 의 Michael Pradel 교수님이 발표했다. 핵심 주장은 생성형 AI 덕분에 코드 변경을 만드는 비용이 크게 줄면서, 병목이 구현에서 검증으로 옮겨가고 있다는 것이었다.

임의의 프로그램 전체가 맞는지 판단하는 건 어렵지만, 변경(delta)은 다르다. 기존 구현의 동작은 이미 대체로 받아들여진 상태이고, 무엇을 바꾸려는지에 대한 자연어 설명도 함께 있다. 둘을 합치면 "의도한 곳만 바뀌고 나머지는 그대로여야 한다"는 부분 명세가 생긴다는 관점이었다. 그래서 앞으로 AI4SE 의 과제는 패치를 만드는 것이 아니라 그 패치가 맞다는 근거를 함께 내놓는 것이라고 정리했다.



구체적인 예로는 자연어로 된 변경 의도를 이용해 동작 회귀를 찾는 Testora, 자연어 산출물에서 패치의 오라클을 추론하는 PatchGuru, 변경을 인식하며 GUI 를 탐색 테스트하는 RippleGUItester 등을 소개했다. 남은 과제로는 오라클 문제, 바뀐 코드를 실제로 실행시키는 문제, 합성 가능성, 그리고 검증 결과를 사용자에게 어떻게 보여주고 코딩 에이전트에 어떻게 자연스럽게 녹일지를 꼽았다.



새로운 이야기라기보다는 이미 체감하고 있던 흐름을 잘 정리해준 발표에 가까웠다. 나도 이제 개발의 중심이 "어떻게 코드를 짤까"에서 "어떻게 검증할까"로 옮겨갔다고 생각하고 있었고, 요즘 동아리 KWEB 의 스터디도 같은 방향으로 운영하고 있다. 그래도 그 흐름을 "변경 단위의 부분 명세"라는 틀로 정리한 건 명세를 다루는 연구를 하는 입장에서 공감이 갔다.

Lost in Automated Specifications: A Time and Space Odyssey in System Testing

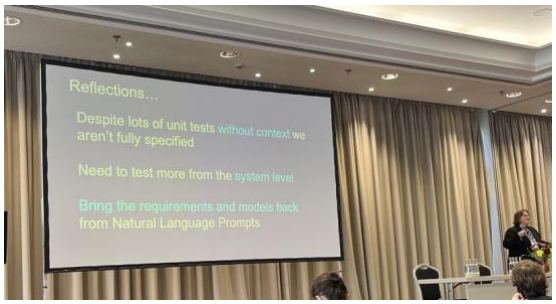
셋째 날 키노트로 Iowa State 의 Myra Cohen 교수님이 발표했다. 시스템 테스팅은 오랫동안 입력, 설정, 동작 순서 같은 것을 모델로 만들 수 있다는 가정 위에 서 있었는데, 이제는 명세가 없거나 불완전하고 시스템은 정의하기 어려울 만큼 복잡해지면서 그 가정이 깨지고 있다는 이야기였다.

가장 기억에 남은 건 과학 소프트웨어 사례였다. 화학 관련 실험을 돌렸을 때 나와야 하는 값이 있었는데, 아무리 다시 돌려도 그 값이 나오지 않았고 한참을 들여다봐도 원인을 몰랐다고 한다. 그런데 도메인 전문가가 보더니 "아, 그건 이것 때문일 거예요"라며 바로 원인을 짚었다는 이야기였다.

이 일화가 키노트의 주제와 맞닿아 있다고 느꼈다. 시스템이 어떻게 동작해야 하는지에 대한 명세가 문서가 아니라 전문가의 머릿속에만 있는 경우가 많고, 그걸 꺼내지 못하면 테스트도 할 수 없다. 자동화된 테스팅이 앞으로 풀어야 할 문제는 결국 이런 암묵적인 명세를 어떻게 다룰지라는 생각이 들었다.

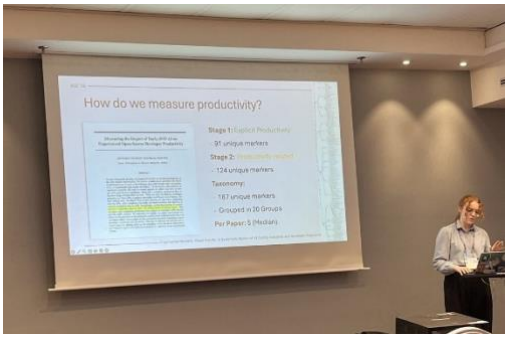


마지막 Reflections 슬라이드도 같은 이야기였다. 유닛 테스트가 아무리 많아도 맥락이 없으면 명세가 완전하지 않고, 그래서 시스템 수준에서 더 테스트해야 하며, 자연어 프롬프트 속에 묻혀버린 요구사항과 모델을 다시 꺼내와야 한다는 것이다.



Fragmented Markers, Mixed Results: A Systematic Review of AI Coding Assistants and Developer Productivity

제목만 보고 이번 학회에서 가장 기대했던 발표였고, 우리 논문과 함께 Distinguished Paper Award 를 받은 논문이다. Leipzig 대학 연구진이 AI 코딩 어시스턴트가 개발자 생산성에 미치는 영향을 다룬 연구 45 편을 체계적으로 리뷰했다.



발표는 숙련된 오픈소스 개발자가 AI 도구를 쓰면 오히려 작업 완료 시간이 19% 늘었다는 METR 의 연구를 예시로 시작했다. 같은 시기에 시간이 크게 줄었다는 연구도, 변화가 없다는 연구도 있는데, 과제도 참가자도 환경도 전부 달라서 서로 모순되는 게 이상하지 않다는 것이 문제 제기였다.

결론은 의외로 "학계도 아직 모른다"에 가까웠다. 연구들이 생산성을 측정하는 지표가 무려 187 가지로 흩어져 있는데, 논문 하나는 보통 그중 5 개 정도만 쓴다. 같은 "작업 시간"이라도 초

단위로 재기도 하고, 설문으로 묻기도 하고, 인터뷰로 정리하기도 해서 연구끼리 비교가 거의 안 된다.



더 흥미로운 건 결과의 방향이 측정 방식에 따라 달라진다는 점이었다. 전체적으로는 긍정적인 결과가 63%로 많아 보이지만, 저자의 경향 서술로 보고된 결과는 81.5%가 긍정인 반면 통계적으로 검증된 결과는 53.9%만 긍정이었다. 체감 지표(71.1%)가 객관 지표(54.5%)보다 긍정적인 것도 같은 맥락이다. 또 생산성 지표는 대체로 긍정인데 AI 도구 자체에 대한 평가는 오히려 부정적인 경우가 많았다. 개발자는 빨라졌다고 느끼면서도 도구는 못 믿는다는 뜻이다.

다만 솔직히 아쉬움이 더 컸다. 내가 기대한 건 "AI 시대의 개발 생산성은 이렇게 재야 한다"는 지표 그 자체였는데, 이 논문은 그 지표를 연구한 연구들에 대한 메타 연구였다. 결국 "다들 다르게 재고 있어서 아직 모른다"는 결론을 정교하게 확인해준 셈이다.

재미있었던 건 발표자도 같은 지점에서 좌절했다고 털어놓은 것이다. 질의응답에서 "의학에서는 연구마다 측정 기준이 다 달라서 결과를 비교할 수 없는 일은 없는데, 왜 우리 분야는 생산성을 어떻게 잴지조차 합의를 못 하냐"는 질문이 나왔다. 발표자는 원래 "어떤 지표에서는 AI 가 효과 있고 어떤 지표에서는 아니다"를 말해주는 연구를 하고 싶었는데, 분석해보니 바로 그 비교가 불가능해서 본인도 많이 답답했다고 답했다. 생산성은 회사마다, 연구자마다 중요하게 보는 게 달라서 쉽게 하나로 모이지 않는다는 것이다.

현업에서 측정하는 사람은 뭘 참고해야 하냐는 질문에는, 재고 싶은 지표가 정해져 있다면 이 논문의 분류 체계에서 다른 연구들이 그 지표를 어떻게 쟀는지 찾아보고 최소한 기존 연구와 비교 가능하게 설계하라고 답했다. 결국 기대했던 정답은 없었지만, 내가 인턴 때 느낀 막막함이 학계에서도 똑같이 막막한 문제라는 걸 확인한 셈이었다.

그래도 하나는 확실히 얻었다. 체감 지표만 보면 효과를 과대평가하기 쉬우니, 객관 지표와 구분해서 봐야 한다는 점이다. 현업에서 AI 도입 효과를 이야기할 때 바로 써먹을 수 있는 교훈이었다.

GANADI: Uncovering C/C++ OSS Reuse Genealogies via Pivotal Function-Based Clustering to Enhance Supply Chain Security

같은 고려대학교 소속 CCS 연구실 분이 발표하셔서 찾아가 들은 발표다. C/C++ 오픈소스가 어디서 어디로 재사용됐는지 "가계도"를 찾아내고, 이를 취약점 추적에 활용하는 연구였다.

처음에는 "이게 지금까지 안 됐었다고?"라는 생각이 들었는데, 듣다 보니 왜 어려운지 이해가 됐다. Java 나 Node 는 패키지 매니저로 의존성을 명시적으로 관리하니 어떤 라이브러리의 어느 버전을 쓰는지 바로 알 수 있다. 반면 C/C++은 라이브러리 코드를 프로젝트 안에 그대로 복사해 넣는 경우가 많아서, 그 코드가 어디서 왔는지 기록 자체가 남지 않는다. 그래서 원본 라이브러리에 취약점이 발견돼도 그걸 가져다 쓴 프로젝트를 찾기가 어렵다.

매일 npm 으로 JavaScript 생태계를 다루는 입장에서, 의존성이 명시돼 있다는 게 얼마나 당연하지 않은 일인지 새삼 느꼈다.

MiniPoly

발표는 공저자 정웅이가 맡았다. 옆에서 보면서 준비를 정말 많이 했구나 싶었고, 이번 학회에서 들은 발표 중에서도 거의 투톱이었다고 생각한다.

시상식은 뱅퀫 전에 별도 시간으로 따로 진행되어서 사람이 아주 많지는 않았다. 그래도 세계 무대에서 상을 받아본다는 게 신기하고 뿌듯했다.

한 가지 아쉬운 점은 포스터를 열심히 준비했는데, 포스터 세션 분위기가 생각보다 활발하지는 않았다는 것이다. 그나마 기억에 남는 건 워크숍에 참여하신 한 분이 찾아오신 일이다. 본인도 오픈소스 개발자들이 짊어지는 부담을 연구하고 있는데, 특히 core-js 메인테이너 zloirock 의 사례가 무척 흥미로운 데이터였다고 하셨다. 폴리필 라이브러리를 사실상 혼자 유지보수해온 사람의 부담을 우리는 자동화로 덜어주려 했던 셈이라, MiniPoly 가 푸는 문제가 그런 결이라서 재미있다는 말씀이 반가웠다.

Etc.

뱅퀫 음식보다 기억에 남은 건 따로 있었다. 점심시간에 잠깐 빠져나와 학회장에서 20 분 거리에 있는 학센 맛집 Haxnbauer 에서 테이크아웃한 학센 젬멜이다. 겉이 단단한 젬멜 빵에 황금빛으로 바삭하게 구운 껍질째 학센을 두툼하게 썰어 넣고, 치즈와 쪽파를 얹은 단순한 메뉴인데, 이걸 먹으러 뮌헨에 다시 가고 싶을 정도로 감동적이었다.



그래서 다음 날 또 갔는데, 이번에는 부드러운 번에 결대로 찢은 고기와 체다 치즈가 들어간 전혀 다른 모습이 나왔다. 때깔부터 달랐고, 아쉽게도 첫날의 그 감동은 다시 오지 않았다.

맥주를 워낙 좋아해서 독일 맥주도 많이 기대했다. 한국처럼 어디서나 같은 맥주가 나오는 게 아니라 식당마다 내놓는 맥주가 저마다 달랐고, 특히 밀맥주가 새로웠다. 다만 맛 자체는 솔직히 기대 이상은 아니었다. 너무 기대를 해서인지, 한국에서 마시는 수입 맥주와 크게 다르지 않게 느껴졌다. 아직 내 내공이 부족한 탓일지도 모르겠다.