ASE 2025 Report

Seoul, Korea

Seongmin Ko, PLRG@Korea Univ.

17-19 November 2025

Introduction

올해 7월 한국에서 열린 PLDI 학회 이후 Automated Software Engineering (이하 ASE) 학회가 11월 그랜드 워커힐 서울 호텔에서 개최되어 참석하게 되었다. 대부분의 연구실 멤버들이 PL과 SE분야 양쪽에 발을 걸치고 있지만 다들 마음은 PL에 가까워서인지, SE 분야의 evaluation 위주 논문을 긍정적으로 바라보지는 않는 것 같다. 나 또한 마찬가지라 그 이유를 생각해보았는데, 주로 논문을 읽고 공부를 할 때 '결과가 아주 좋음'에 해당하는 논문보다는 '문제 해결을 위한 새로운 방법론을 제시'하는 논문들을 위주로 읽고 공부하며 나에게 지적 재산(?)이 되는 논문만 골라 읽어서 그러한 경향성이 생긴 게 아닌가 싶다. 하지만 우리 연구실은 PL 기술과 SE의 real-world가 합쳐진 연구실이기도 하고, 나도 SE에 가까운 주제를 연구하고 있다. 그래서 ASE에 참석하여 SE 논문은 무엇이 중요한 포인트인지, 어떻게 연구의 방향을 잡아야 하는지 같은 것들을 얻고 싶었다.

이번 ASE는 이전에 참석했던 PLDI와 다르게 논문이 구술 발표에다 전부 필수적으로 포스터 발표까지 진행해야 했다. 홀이 총 7개가 동시에 돌아가기도 하고 발표 시간도 10분으로 짧게 가져가는 대신 포스터를 상시 전시하고 4시쯤부터 6시까지 포스터 세션 집중 시간이었다. 사실 질의 포함 10분 발표로 자신의 논문을 모두 설명한다는 건 불가능하기 때문에 머릿속엔 어딘가 비어있는 정보가 존재한다. 그리고 내가 8분 발표를 듣고 깔끔하게 이해를 할 수 있는 정도의 경험이 없기 때문에 포스터 세션은 정말 좋은 경험이었다. 포스터를 가서 어떻게 했는지, 평가는 어떤식으로 이루어졌는지, 왜 하게 되었는지 동기를 묻는 등 발표자들이 어떤 생각을 가지고 있는지 더 깊게 알 수 있어서 좋았다.

ASE에 출판하여 발표하는 논문들을 보면서 '어려운 문제를 어떻게 잘 풀었는가'에서 '왜 이게 아직 풀리지 못한 문제인가'를 찾아내는 방법을 배울 수 있었다. ASE에 붙어서 발표를 하게 된 성민이(park)와 같이 논문을 작성했던 서진이, 그리고 학회에 갈 수 있게 지원해주시고 학회에서 많이 배울 수 있도록 도와주신 박지혁 지도교수님에게 감사를 전한다.

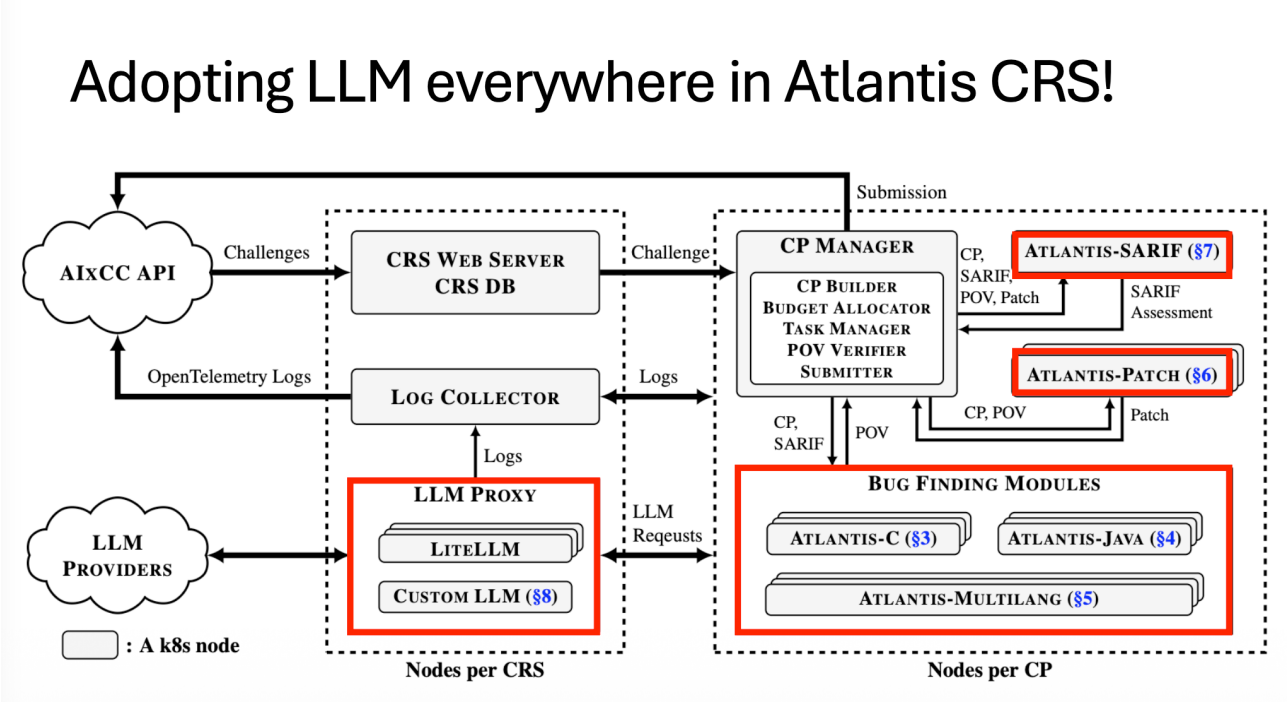
Interesting Talks

Overall

나는 주로 program repair, fault localization, code generation, analysis 주제들을 위주로 봤다. 그 외에 fuzzing 논문들이나 security 쪽은 포스터를 통해서 흥미로운건 몇 개 볼 수 있었다. 이번 ASE 논문의 대부분이 LLM을 이용해서 자기 분야의 성능을 SOTA 이상으로 끌어올린 논문이었다. 이전에는 단순히 'LLM을 이용해봤는데 잘 하더라'의 느낌이라면 지금은 'LLM에게 그냥 맡기지 말고 도움이 되는 특정 정보들을 추가적으로 제공하면 더 잘 하더라'와 같은 논문들이 많았다. 그리고 agent를 통해 더 자동화된 LLM 사용으로 성능을 더 끌어올렸다는 논문이 전체 논문의 절반 이상이었다. 그래서 방법론을 이해하기가 크게 어렵지 않았다. 그래서 평가를 어떤 방식으로 했는지, 자기들이 어느 부분에서 강점이 있고 그걸 살리기 위해 어떤 평가를 선택했는지 등을 위주로 봤으며 연구 발표는 어떤 흐름으로 해야하는지에 좀 더 집중해서 발표를 들었다.

Hyperscale Bug Finding and Fixing: DAPRA AIxCC and Team Atlanta

수요일 오프닝 키노트로 Georgia tech의 김태수 교수님께서 AIxCC를 우승하게 된 여정을 발표해주셨다. AIxCC란 AI Cyber Challenge로 infrastructure의 취약점을 찾고 패치하는 자동 시스템을 구축하여 보안취약점을 개선하는 대회로 23년 8월에 공지되어 2년간 진행된 대회이다. 아무래도 기반 라이브러리의 취약점을 찾는 대회이기 때문에 대규모 코드에 대한 취약점 탐지 기술이 필요하고, 각 challenge별 취약점들의 특성이 다른 점들을 반영해야하는 어려운 대회였지만 AI를 활용해서 그러한 문제를 해결할 수 있을까?가 대회의 취지였는데 team Atlanta는 AI를 기가막히게 활용해서 많은 취약점과 패치를 자동으로 잡아 우승하셨다.



사실 AI에 대한 논문이 아닌 다른 domain에 AI를 활용하여 문제를 해결하는 논문들을 읽어보면 평가가 좋은건 맞지만 novel한 방법론을 사용하는 것도 아니고, 재현이 불가능한 경우가 많다는 점들이 많아 실망스러운 경우가 많았다. 하지만 team Atlanta의 전체 구조를 보니 'AI를 쓸거면 이렇게 써야지'라고 생각이 들었고 AI를 활용한 논문들이 겉보기엔 작은 문제들에 특화되어 이게 맞나 싶은 생각이 들겠지만 그것들이 모이면 하나의 큰 문제 해결 모델이 될 수 있다는 것을 느꼈다. 사진에서 보다시피 각 문제에 특화된 9개의 LLM agent를 만들고 또 각 agent들이 유기적으로 정보를 공유하며 서로를 보강해주는 구조를 만들었다. 2년동안 해당 모델을 돌리는데 사용된 비용(컴퓨팅 비용 및 LLM API 비용)은 총 359,000 달러로 한화로 자그마치 5억원의 비용이 발생했다고 한다. 이렇게 보니 정말 real-world에서 사용될만한 대형 아키텍처에서는 LLM의 cost를 줄이는 것도 큰 의미가 있겠다 싶었다.

LSPFUZZ: Hunting Bugs in Language Servers

이 연구는 코드 에디터에서 코드를 짤 때 여러 기능을 제공해주는 LS (Language Server)의 bug를 잡는 fuzzer에 대한 내용이다. 해당 연구가 LSP에 특화된 fuzzer를 어떻게 만들고 어떤 방법론을 썼는지에 대해보다 이 연구에 대한 동기와 실제 사용 사례가 더 크게 와닿은 연구였다. 물론 LS를 fuzzing하려면 입력으로 사용자와의 상호작용(클릭, 호버링 등)이 있기 때문에 기존의 fuzzing 기법을 단순히 적용하면 좋은 효과를 내지 못하는 것을 해결 한 것은 좋은 문제와 좋은 해결이라고 생각한다. 저자는 vs code를 사용해서 clangd 언어 코딩을 하던 중 미완성된 function call을 사용하여 assignment를 하는 코드의 '=' 기호를 호

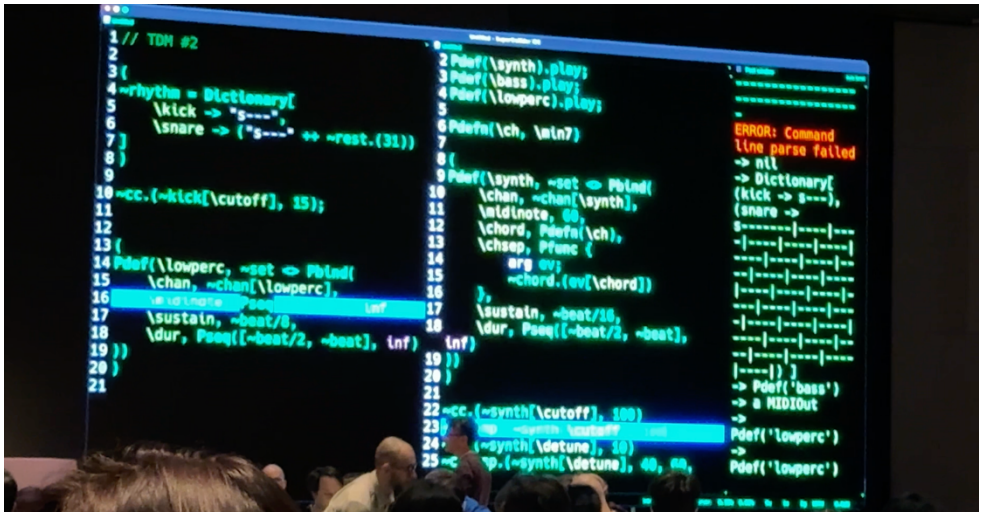
버링하면 LSP 서버가 crach가 나는 것을 발견하였다. 처음에는 LSP 서버가 자꾸 터져서 불편함을 겪다 원일을 찾아보자 해서 delta debugging 방식으로 원인을 찾게 되었다고 한다. 이러한 불편함을 해결하기 위해 LSP에 특화된 fuzzer를 개발하여 여러 버그들을 찾아 제보하게 되었다. 그 결과 동기를 제공했던 clangd의 LSP 서버의 경우는 보안에 매우 취약한 프로그램으로 판단되어 unstrusted workspace에 대해 disable하기로 결정되었으며 또한 sorbet 언어의 공식 테스터로 활용되고 있다고 한다. 연구를 진행하다 보면 '문제를 어떻게 해결할 것인가'보다 '어떤 문제를 해결할 것인가'가 더 중요하다고 느낄 때가 많다. 내가 문제라고 생각하는 것이 정말 문제가 맞는가?하는 의심이 들 때마다 해당 연구 저자의 motivating example 처럼 실제로 내가 겪어서 문제가 되었고 불편함을 느낀 동기가 있으면 그런 동기부여에 대해서 자기의심을 덜 할 것 같다는 생각이 들었다. 그렇게 실제로 경험한 문제에서 시작했다 보니 그 결과로 실존하는 문제를 해결했다는 점을 자랑스럽게 내세울 수 있다는 점이 많은 생각을 들게 하는 발표였다.

Etc.

우선 워커힐 서울 호텔의 환경이 매우 좋았다. 학회 도중 나오는 케이터링과 학회장 환경, 창가로 보이는 한강 풍경 등 환경적인 부분은 정말 최고였다. 리셉션이나 뱅퀫으로 제공된 식사는 참석한 다른 친구들이 올릴테니 나는 가장 많이 먹었던 곶감을 올리려고 한다. 너무 맛있어서 학회장을 오면 항상 제일 먼저 하나씩 챙겨먹었다. 확실히 환경이 좋으면 머리가 맑아지는 효과가 있어서인지 학회 발표에 좀 더 집중을 잘 하게 된다.

케이터링 – 곶감은 오른쪽에서 5번째

그리고 학회 행사 중 가장 충격적이었던 것은 월요일 리셉션 시간동안 진행된 라이브 코딩 디제잉이었다. 이틀차 뱅퀫 시간에 진행된 국악 공연 또한 좋았지만 충격적인 면에서 라이브 코딩 디제잉이 압승이었다. VIM을 켜서 미리 정의된 명령어로 비트를 찍고, 코드를 찍어가며 백그라운드 비트와 멜로디를 쌓아가는게 신기하기도 했고 전공자들이다 보니 모두가 저 언어는 어떻게 정의했을까, 컴파일하고 실행은 어떻게 되는걸까 등 기술적인 이야기를 하고 있는게 너무 웃겼다. 그리고 디제이 분도 실수하시는 parse failure가 간간히 등장해서 인간미적인 모습을 보여주셨다.



리셉션 동안 진행되었던 라이브 코딩 디제잉!

워커힐 서울 전경