

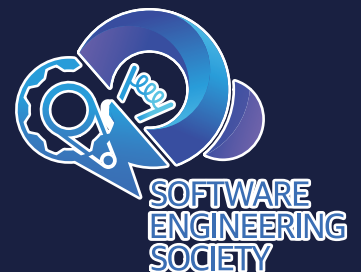


COMMUNICATIONS

OF SOFTWARE ENGINEERING SOCIETY

09/2026 VOL.6 NO. 2

소프트웨어공학 소사이어티 소식



CONTENTS



COMMUNICATIONS OF SOFTWARE ENGINEERING SOCIETY 소프트웨어공학 소사이어티 소식

- 03 편집자 인사말 / 이주용 (UNIST)
- 04 기고문
스마트팩토리 협동 로봇을 위한 예지 보전 요소 기술 / 이정원 (아주대)
- 14 신진연구자 소개
김기섭 (DGIST)
김형석 (충남대)
- 20 Top Paper
버그 수를 믿어도 될까? 퍼져 벤치마킹 결과의 무작위성 측정에 관하여/ 이성민 (UCLA)
- 23 국내외 학술행사 소개
ICST 2026 개최 후기 / 김문주, 홍신
ICSE 2026 참가 후기 / 김영재, 한승현 (UNIST)
제14회 소프트웨어공학 단기전문가강좌를 마치고 / 지은경 (KAIST)
제14회 소프트웨어공학 단기전문가강좌 수강 후기 / KHAREL AROGYA (아로겨) (KAIST)
- 28 기업탐방 특집 - 솔루션링크
AI 시대, 소프트웨어공학의 가치를 다시 묻다
- 33 소사이어티 광장
- 34 소사이어티 조직도
- 34 발행정보

09/2026
VOL.6 NO. 2



• 발행인 백종문 • 편집인 강종구, 김기섭, 마유승, 이주용 • 발행일 2026. 9. 1



이주용
UNIST 교수

안녕하세요. UNIST 이주용입니다. 소식지를 통해 인사드리게 되어 반갑습니다.

혹시 모르실 분들을 위해 가장 기본적인 소식부터 알려드립니다. KAIST 백종문 교수님이 제20대 소프트웨어공학 소사이어티 회장으로 취임하셨으며, 이에 맞추어 새로운 회장단 및 운영위원진이 활동을 시작하였습니다. 조직 구성에 대한 자세한 사항은 이번 소식지의 뒤쪽에서 확인하실 수 있습니다.

이번 소식지는 새로운 회장단에서 발간하는 첫 번째 호입니다. 이번 호는 다음과 같이 구성되어 있습니다.

- 직전(제19대) 회장을 역임하신 아주대학교 이정원 교수님께서 “스마트팩토리 협동 로봇을 위한 예지 보전 요소 기술”에 대한 기고문을 보내주셨습니다.
- 신진연구자 두 분과의 인터뷰: DGIST의 김기섭 교수님, 충남대학교의 김형석 교수님과의 흥미로운 대화가 준비되었습니다.
- “버그 수를 믿어도 될까? 퍼저 벤치마킹 결과의 무작위성 측정에 관하여”라는 다소 도발적인, 하지만 통찰력이 깊은 글을 UCLA에서 박사후 연구원으로 활동 중인 이성민 박사가 보내 주셨습니다.
- 지난 5월에 성공적으로 개최된 ICST 개최 후기를 KAIST 김문주 교수님과 충북대 홍신 교수님께서 보내 주셨습니다.
- UNIST의 김영재, 한승헌 학생이 ICSE 2026 참관 후기를 보내 주셨습니다.
- 올해 여름 단기 강좌는 “AI 증강 명세 주도 개발과 하네스 엔지니어링”을 주제로 열렸습니다. 소사이어티 부회장이신 KAIST 지은경 교수님이 개최 후기를 보내 주셨습니다.
- KAIST Kharel Arogya 학생이 단기 강좌 수강에 대한 소회를 질의 및 응답 형식으로 보내 주셨습니다.
- 소프트웨어 공학 전문 기업이자 소사이어티의 오랜 후원 기업인 솔루션 링크에 대한 기업 탐방문이 실렸습니다.
- 마지막으로, 소사이티 회원분들의 소식을 실었습니다. 나누시고 싶은 소식이 있으시면 언제든지 편집부로 연락주세요.

알차게 소식지를 꾸며 주신 모든 저자분들께 감사드립니다. 그리고 저와 함께 소식지를 준비해 주신 ETRI의 마유승 박사님, 성신여자대학교의 강종구 교수님, DGIST의 김기섭 교수님께도 감사드립니다.

그럼, 즐겁게 소식지를 읽으시기를 바라며, 이만 줄입니다.

2026년 9월

이주용 드림



스마트팩토리 협동 로봇을 위한 예지 보전 요소 기술



아주대학교 이정원 교수

1. 서론

스마트팩토리는 설계 및 개발, 제조, 유통 등의 생산과정에서 인공지능 기반의 자동화 솔루션이 결합된 지능형 생산공장을 말한다. 구체적으로는 공장 내 설비와 기계에 사물인터넷(IoT) 기술이 결합되어 공정 데이터를 실시간으로 수집하고, 모니터링 소프트웨어를 통해 지속적으로 추세를 관찰하면서 데이터 및 주요 이벤트를 분석해 스스로 제어할 수 있는 형태이다. 지금까지의 공정 자동화 기술은 공정별로만 자동화가 이루어져, 전체 공정을 유기적으로 관리하기 어려웠다. 하지만 IIoT(Industrial Internet of Things) 기술 개발로 인해[1], 모든 설비가 무선 통신으로 연결되고, 전후 공간 데이터를 자유롭게 연계할 수 있어 스마트팩토리의 실현이 가능해졌다. 이를 통해 스마트팩토리에서는 최적의 생산 환경 구축을 목표로, 기기 이상에 의한 생산 라인 중단과 품질 리콜과 같은 치명적인 상황에 사전 대응할 수 있는 환경을 구축한다. 따라서 설비 기기의 결함 예측, 고장 진단을 수행하는 예지 보전(Predictive Maintenance, PdM) 기술이 중요해지고 있다[2].

한편, 스마트팩토리에서는 전 생산과정을 무인화하기 위한 공장 자동화와 달리, 반드시 인간이 제조과정에 참여해야 하는 섬세하고 반복적인 작업에 협동 로봇을 투입한다. 협동 로봇(Cobot: Collaborative Robot)은 작업자와 같은 공간에서 물리적인 상호 작용을 수행할 수 있도록 높은 안전성을 갖춘 산업용 로봇을 말한다. 기존 산업용 로봇의 경우, 이미 설치된 로봇을 다른 작업에 투입할 때 많은

오버헤드 문제(예: 전문가에 의한 프로그램 재설정과 생산라인 리모델링 등)를 발생시켰다. 이와 같은 문제점을 해결한 협동 로봇은 작업자가 수행하던 반복적인 작업을 직접 교시하거나 쉽게 프로그래밍하여 자유롭게 모션 명령을 지시할 수 있다. 또한, 가벼운 충돌도 감지할 수 있어, 충돌 감지 시 바로 멈췄다가 작업을 재 시작할 수 있으며, 이로 인해 인간과의 협력이 가능하고 생산성을 향상시킬 수 있다[3]. 다음 그림 1은 스마트팩토리에 도입된 협동 로봇의 활용시 장점을 예시한다.

협동 로봇은 미리 정해진 고정적인 작업이 아닌 프로그램 가능한 모션(Programmable Motion)에 의해 구동되는 작업을 수행한다. 출하 시점에 다양한 테스트(예: 조립 테스트, 내구성 테스트, 캘리브레이션 등)를 통과한 협동 로봇을 공장에 납품하면, 각 공장에서는 작업의 종류, 사람과의 협력 과정(작업공간 크기 및 범위, 모션 순서 등), 작업에 필요한 엔드이펙터(예: Gripper, Vacuum Pump, Presser 등)에 따라 사용자가 필요한 작업을 직접 프로그래밍하여 협동 로봇에게 지시한다. 보통, 한 대의 협동 로봇은 하루에도 2~3개 이상의 서로 다른 프로그램을 실행할 수 있으며, 실행되는 프로그램도 수십에서 수백 개의 모션 명령어 조합으로 이루어진다. 그림 2는 협동 로봇이 차량용 매트 제작하는 공정에 투입되어 각 단계별로 컨베이어 벨트에 매트 재료를 로딩하고 펼치고, 자르고, 압착한 뒤, 바느질 기계에 투입 및 이동하는 과정을 반복하면서 실리콘으로 마무리하는 작업을 예시하고 있다. 협동 로봇이 여러 단계에 투입되어 인간과 같이 다양한 작업을 하고 있으며, 이때 예지 보전 관점에서 로봇의 모션과 센싱 데이터를 수집하여 작업을 제대로 수행하고 있는지, 결함은 없는지 분석하는 과정과 지속적인 상태 모니터링 과정이 필요하다.

본문에서는 전반부에 스마트팩토리의 예지 보전 기술 및 협동 로봇의 결합 진단 프로세스를 소개하고 후반부에는 협동 로봇 예지 보전을 달성할 수 있는 상태 모니터링 소프트웨어 설계를 제시한다.

2. 예지 보전(Predictive Maintenance)

2.1 유지보수 전략

스마트 팩토리의 유지 보수 전략은 성숙도 수준에 따라 크게 4가지로 분류된다[5].



그림 1. 협동 로봇의 활용 [4]

	(Phase 1) Fabric loading	(Phase 2) Fabric cutting	(Phase 3) Fabric sewing	(Phase 4) Fabric gluing	(Phase 5) Mat sewing	(Phase 6) Rubber attaching	(Phase 7) Mat packing
Task	- Stretched fabric loading - Stretched fabric transfer	- Stretched fabric cutting - Cut fabric transfer	- Cut fabric placement - Logo lockstitch - Sewn fabric loading - Sewn fabric transfer	- Sewn fabric placement - Glue spread - Sewn fabric attach - Attached fabric loading - Attached fabric transfer	- Attached fabric placement - Edge sewing - Sewn mat loading - Sewn mat transfer	- Sewn mat placement - Front rubber inserting - Sewn mat flip - Rear rubber inserting - Mat loading - Mat transfer	- Box molding - Mat placement - Mat packing
Related device	- Single/Multiple cobot - Conveyor belt	- Presser - Conveyor belt	- Multiple cobot - Sewing machine - Conveyor belt	- Multiple cobot - Conveyor belt	- Single/Multiple cobot - Conveyor belt - Sewing machine	- Multiple cobot - Presser - Conveyor belt	- Multiple cobot - Welding machine

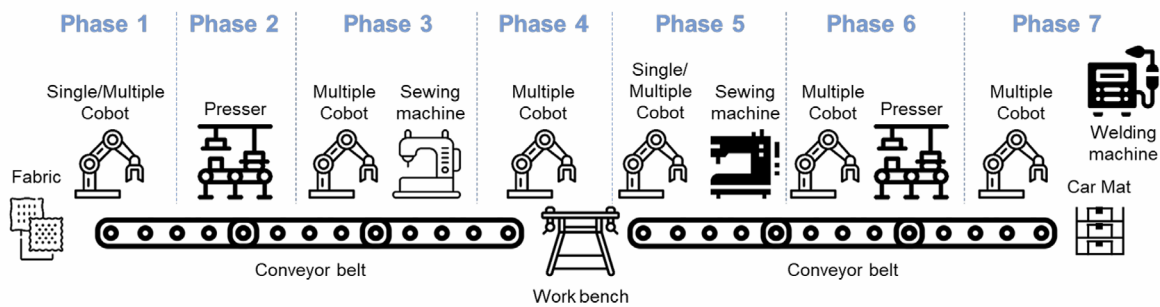


그림 2. 협동로봇의 프로그래머블 특성 (예: Car Mat 제작 과정)

- **Level 1 – Visual inspection:** 주기적으로 장비의 물리적 검사를 수행하며, 유지 보수 전략은 검사관의 전문 지식에만 기반함
- **Level 2 – Instrument inspection:** 주기적으로 장비의 물리적 검사를 수행하며, 유지 보수 전략은 검사관의 전문 지식과 계측 장비의 판독값의 조합에 기반함
- **Level 3 – Real-time condition monitoring:** 장비에 대한 지속적인 실시간 모니터링을 수행하며 사전 설정된 규칙 또는 심각도 수준에 따라 경보가 발령됨
- **Level 4 – PdM 4.0:** 장비에 대한 지속적인 실시간 모니터링을 수행하며 회귀분석, 기계학습, 딥러닝 등과 같은 예측 기법을 기반으로 경보가 고장전에 발령됨

이와 같이 예지 보전(PdM) 기술은 스마트 팩토리 RCM(Reliability Centered Maintenance) 전략의 가장 최신화된 기술로써, 고장이나 결함의 원인을 예측·제거함으로써 기기의 효율적인 유지보수를 수행하기 위한 기술이다. 설비가 고장 난 후에 이를 보수하는 형태인 ‘사후정비(RM, Reactive Maintenance)’ 방식을 적용하였으나, 설비 고장시 손실이 매우 크고, 설비를 복구하여 재가동하는 데 많은 시간이 소요된다. 이를 개선하기 위한 ‘예방정비(PM, Preventive Maintenance)’ 방식은 사후보전 방식의 문제점을 개선하고, 설비가 고장 나기 이전에 주기적인 보수를 시행하는 방식이다[6-7]. 그러나 이와 같은 방법 역시, 교체 시기 이전에 먼저 손상이 발생하거나 결함을 감지하지 못한 경우 심각한 설비 고장으로 이어지기 때문에 문제가 된다. 또한, 통계적 기준에 따라 부품을 교체하기 때문에 과도한 보전이 될 수도 있다. 따라서 설비의 초기 결함 단계부터 설비를 관리하는 예지 보전(PdM) 방식을 채택하기 시작하였다.

예지 보전 기술은 설비의 초기 결함, 즉 이상 징후가 나타나는 시점부터 설비에 대한 평가와 분석을 시행하여 미리 결함을 제거한다. 최근에는 인공지능 기술의 접목으로 높은 성능의 결함 예측이나 고장 진단과 같은 예측 정비가 가능해졌으며, 스마트 팩토리의 핵심 기술로 급부상하고 있다[8].

2.2 예지보전 접근 방식

예지 보전 기술에 접근하는 방식은 크게 두 가지 관점(도메인 지식수준, 데이터 양)에 따라 네 가지 접근 방식으로 나누어진다[9]. 먼저, 도메인 지식도 적고 데이터도 적은 상황에서는 규칙 기반(Rule-based)의 진단 방식을 활용한다. 이와 같은 방식은 정의된 규칙에 대한 종속성이 높아 실데이터에 대한 진단율이 낮다. 반면, 데이터는 적으나 도메인 지식은 많은 상황에서는 기기의 물리 모델(Physics-based)을 설계하여 내부 상태를 진단한다. 물리 모델의 경우, 도메인에 대한 깊이 있는 모델링을 통해 진단 성능이 좋으나 다양한 환경 변수를 고려하기 어려워, 실제 필드 적용에는 한계점을 갖고 있다. 그러나 데이터의 양이 많아지게 되면, 다양한 환경 조건을 고려할 수 있고 실제 데이터 패턴을 반영할 수 있어 필드에서 적용성이 높다. 이때, 도메인 지식수준에 따라 단순 데이터 중심의 방법(Data-driven)과 물리 모델을 결합하는 하이브리드(Hybrid) 방법으로 구분된다. 또한, 데이터 중심의 방법은 세부적인 적용 기술에 따라 통계 기반(Statistic-based)의 방법과 학습 기반(Learning-based)으로 구분될 수 있다.

3. 협동 로봇의 예지 보전

3.1 결함진단 프로세스

협동 로봇 결함 진단 과정에서 필요한 데이터는 다음 그림 3과 같이 고장(Failure), 결함(Fault), 이상성(Anomaly), 정상(Normal) 데이터로 나누어 정의할 수 있다. 이때, 고장, 결함, 이상성 정의는 IEEE 표준(IEEE Std 1856-2017)을 따른다[10].

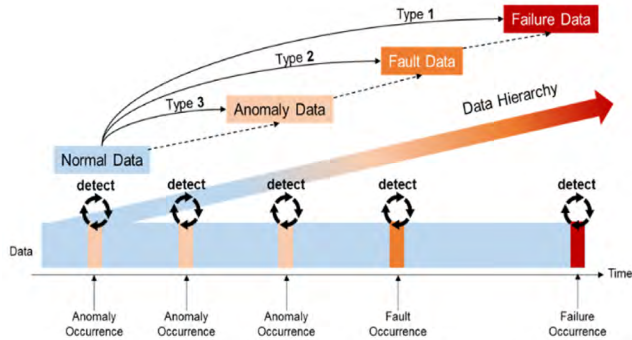


그림 3. 결함 진단 과정에서 생성되는 데이터

- **Failure(고장):** 정상상태를 넘어가는 편차가 수용 불가능한 수준
- **Fault(결함):** 이상이나 고장 상태의 원인
- **Anomaly(이상):** 정상상태를 넘어가는 편차가 수용가능한 수준

이와 같은 정의를 보면, 결함은 고장의 원인을 제시할 수 있으며, 여러 결함이 조합되어 하나의 고장의 원인이 될 수 있음을 의미한다. 또한 이상은 결함까지는 아니지만, 정상 상태를 벗어난 빈도와 패턴을 분석하여 고장을 예측할 수 있는 전조 증상으로 판단할 수 있는 근거가 된다. 이를 고장 진단의 목적으로 보고 세분화하면 다음과 같다.

- **Failure Prediction:** 고장에 직·간접적으로 연관이 있는 데이터를 수집하여 건전성(Health Prognostics) 평가, 내장된 각종 부품의 고장 상태를 분석하여 잔존 수명을 예측하는 기술로 결함 사전 예측, 건전성 평가, 수명 예측 등이 포함됨
- **Fault Detection:** 부품의 물리적인 문제(예: 기어 마모-볼트 결합력 약화 등)를 검출
- **Fault Localization:** Fault가 어느 부품에서, 또 어떤 원인으로 발생했는지 추적 가능함
- **Fault Diagnosis:** Fault Detection과 Localization 단계를 합하여 고장을 탐지하고 원인을 제시할 수 있음
- **Anomaly Detection:** 정상 데이터로부터 큰 차이를 보이는 이상 구간을 검출하며, 각 결함에 영향을 미치는 이상성간의 관계를 찾아냄

다시 말해서 생성·수집되는 데이터 관점에서 볼 때, 고장 시점 이전에 결함을 예측하기 위해서는 빈번한 결함 패턴을 발견하고, 각 결함에 대한 전조 증상으로서 이상 데이터를 조기에 파악하는 게 매우 중요하다

3.2 협동 로봇의 프로그래머블 특성

스마트 팩토리의 핵심 장비인 협동 로봇은 생산과정의 무인화 또는 일부

반복적·기계적인 작업의 자동화 등을 위해 활발히 도입되고 있다. 따라서 효과적인 유지 보수를 위해서는 협동 로봇의 작업 과정과 결함 등에 대한 정보를 수집하고 모니터링 하면서 전사적인 관리가 필요하다. 협동 로봇을 구동하는 프로그램은 현장에서 작업자에 의해 직접 코딩 되기도 하고 작업 교시에 의해 자동 생성되기도 한다. 프로그램을 구성하는 각 모션은 회전 반경이나 속도, 모션의 시점 및 종점(위치 좌표의 변화) 등에 따라 매우 복잡한 인자들의 조합으로 구분될 수 있다. 다음 그림 4는 협동 로봇의 프로그래머블한 특성으로 인한 결함 진단의 어려움을 예시한다[11].

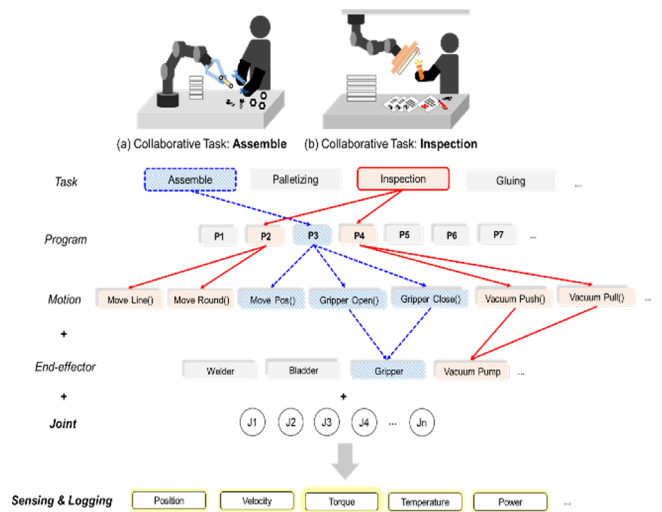


그림 4. 협동 로봇의 프로그래머블 특성 분석

협동 로봇이 수행할 수 있는 작업(그림 4(a) 조립(Assemble), 그림 4(b) 검사(Inspection))에 따라 수반되는 프로그래머블 모션의 계층 구조를 예시하고 있다. 하나의 작업은 여러 개의 프로그램(그림 4(b)의 경우, P2와 P4)에 의해 완성될 수 있고, 하나의 프로그램(그림 4(a), P3)이 반복적으로 호출되어 완성될 수도 있다. 또한, 하나의 프로그램은 복수개의 모션이 조합되어 작업을 수행하게 된다. 예컨대, 조립 작업 P3의 경우, 세 가지의 모션 명령어 Move_Pose(), Gripper_Open(), Gripper_Close()가 존재하며, 작업에 사용되는 엔드 이펙터의 종류에 따라라도 무수히 조합되어 생성될 수 있다. 이때, 센서가 장착된 축(J1 ~ Jn)별로 다양한 작업(Task), 프로그램(Program), 모션(Motion), 엔드이펙터의 설정에 따라 센싱되는 데이터 패턴이 다르므로, 프로그래머블 모션 실행 조건에 따라 진단 기준이 설정되어야 하는 어려움이 있다.

3.3 고장 예측

공장 내의 기계시스템들은 하중이 계속 작용하는 환경에서 마모(Wear), 균열(Crack), 성능 저하(Degradation) 등의 원인으로 인해 고장(Failure)이 필연적으로 발생하게 된다. 그러므로 다양한 결함에 대해 제대로 관리하지 못하여 한계 수준을 넘어가게 되면, 치명적인 사고나 사용 중단 사태가 발생하고, 막대한 피해와 비용 손실이 발생한다. 이를 막기 위해, 로봇 시스템의 이상을 조기에 감지하고 고장을 예측하는 고장 예측하는 기술들이 연구되고 있다.

고장 예측 연구는 고장에 직접 혹은 간접적으로 영향을 받은 데이터를 수집하여 건전성을 평가하는 방법이나, 로봇에 내장된 각종 부품의 고장 상태를 분석하여 잔존 수명을 예측하는 방법에 집중되고 있다. 예를 들어, [12]에서는 모터 회전자에 인코더 신호를 사용하여 산업용 로봇의 고장 진단 및 건전성 평가 방법을 제안하였으며, [13]에서는 비전 센서 데이터를 기반으로 TCP(Tool Center Point)의 위치를 매칭하는 과정을 통해 로봇의 건전성 저하를 평가하는 방법을 제안하였다. 이와 같은 연구에서는 직접적인 고장에 연계되지는 않지만, 고장을 유발할 수 있는 결함 상황을 검출하는 방법을 제안하고, 이를 통한 건전성 평가 결과를 제시하였다. 반면, [14]에서는 결함 상황에 직접 대응하여 연구를 수행하였다. 결함을 지닌 부품을 로봇 팔의 손목에 직접 장착하여, 정상 손목을 장착한 로봇 팔의 토크 데이터를 비교 분석하였는데, 결과적으로 결함이 있는 손목을 장착한 로봇은 건강한 관절에 비하여 약 두 배 이상의 전력 소모량을 보임을 확인하였다.

또한, 로봇에 내장된 모터의 잔존 수명(RUL, Remaining Useful Lifetime)을 예측하기 위한 연구들도 진행되고 있다. [15]에서는 통계 기반의 방법으로 칼만 필터(Kalman Filter)를 활용하여, 베어링 잔존 수명 추정 모델을 제안하였다. 또한, 최근에는 학습을 기반으로 한 예측 모델도 존재하는데, 순환 신경망(RNN, Recurrent Neural Network)[16], 깊은 신경망(DNN, Deep Neural Network)[17], 앙상블 학습(Ensemble learning)[18] 기반의 잔존수명 예측 모델이 제안되었다. 구체적으로는 측정된 신호 데이터(예: 진동 신호, 토크 신호 등)와 잔존 수명 데이터를 가지고 연관성을 분석하는 방식이다. 그러나 이와 같은 기존의 연구들은 고장 패턴 분석을 수행할 수 있다는 장점은 지녔으나, 고정된 운행 환경에만 적용할 수 있어 협동 로봇과 같이 투입된 공장마다 서로 다른 프로그램을 운영하는 경우 적용할 수 없다. 이러한 문제점을 해결하기 위해서는 센서 데이터를 다양한 운행 환경의 조건(예: 실행된 프로그램, 모션 등)을 기준으로 연계하여 분석할 수 있어야 하며, 운행 조건에 따른 이상성 판단 기준도 상황에 맞게 정립되어야 한다.

하지만 출하 시점에는 협동 로봇이 수행할 수 있는 모든 프로그래머블 모션에 대한 테스트를 수행할 수 없다. 산업용 협동 로봇은 ISO/TS 15066 표준[19]을 기반으로 위험성 및 안전성, 성능에 대해 고려해야 하는데, [20]에서는 [19] 표준에 따른 사람과 로봇 간의 안전한 협력 지침을 제시하고 있다. 또한, [21]의 연구에서는 협동 로봇이 인간과 함께 수행하는 'welding' 작업에 초점을 맞추어, 작업 셀 안에서 협동 로봇의 성능 평가를 통한 기기의 상태를 파악하기 위한 연구를 시도하였다.

이때, 산업용 협동 로봇 테스트는 기기의 건전성이나 성능을 평가할 뿐만 아니라, 발생된 결함 혹은 고장의 원인(예: 무리한 프로그램의 가동, 과부하 등)을 파악하는 것에 목표를 둔다. 그러나 기존에 수행된 연구에서는 일부 작업이나 상황(예: Welding Cell, Assembly Cell 등)에 국한되어 테스트를 수행하였을 뿐만 아니라, 단순 결함이나 고장을 진단하는 것에 초점을 맞추고 있어 결함이나 고장의 원인(예: 과부하, 무리한 프로그램 구동 등)을 파악할 수 없다. 따라서 고장이 진단된 시점에 원인을 제공한 모션과 프로그램을 추적하여, 진단의 원인을 규명할 수 있는 체계가 필요하다.

3.4 결함 진단

협동 로봇의 결함은 로봇의 부품에 물리적인 문제(예: 기어 마모, 볼트 결합력 약화 등)를 지니고 있어, 로봇의 고장을 일으킬 수 있는 문제를 의미한다. 보통, 협동 로봇은 산업용 로봇 팔(Industrial Robot Arm)에 속하기 때문에, 위치, 속도, 토크 등의 다양한 센서로부터 신호가 감지하여 결함을 진단하게 되는데, 산업용 로봇과 다르게 협동 로봇의 경우 사용자에 의해 설계된 다양한 작업 환경에 따라 축별 구동 방식이 매우 복잡하다는 문제점이 있다. 이로 인해, 협동 로봇의 결함 검출은 결함 진단을 축소하여 로봇을 구성하는 부품 단위로 진단하는 방법이 일반적이다.

예를 들어, 모터[22-23], 롤링 베어링[24-26], 기어[27-29], 센서[30]의 결함 진단 관련 연구에서는 통계적으로 결함 발생 확률 분석을 수행하여 각 기기의 건전성 인자(예: 진동(Vibration) 신호, 속도(Velocity) 신호, 운동량(Kinetic) 신호 등)을 도출하고, 수립된 모델에 대한 해석을 위해 물리 공간 분석(예: 주파수 도메인 분석, 시공간 분석 등)을 수행한다. 최근에는 결함 진단을 수행하기 위해 선형 판별 분석(LDA, Linear Discriminant Analysis)[31], 서포트 벡터 머신(SVM, Support Vector Analysis)[32], 극학습 기계(ELM, Extreme Learning Machine) [25-26], 심층 신경망(CNN, Convolutional Neural Network)[33], 순환 신경망(RNN, Recurrent Neural Network) [16]과 같은 딥러닝 모델을 적용한 사례도 존재한다. 그러나 이와 같은 부품 단위의 진단은 로봇 형태로 조립되어 운행되는 경우, 적용하기 어려워 조립 이전의 개발 단계에 적용이 가능하다는 한계점이 있다.

또한, 결함의 진단 범위를 확장하여, 컴포넌트(Component), 파트(Part) 단위의 진단 역시 존재한다. [23]에서는 산업용 로봇의 진동 신호를 기반으로 기어박스의 결함을 진단하는 방법을 개발하였다. 특히, 산업용 로봇의 기어박스에서 취득되는 진동 신호로부터 결함에 대한 정보를 극대화하고 정량화하는 건전성 인자를 정의하였다. 그러나 개발된 건전성 인자는 산업용 로봇의 운행 환경(예: 작업 종류, 프로그램 및 모션 종류, End-effector의 종류, 로드 무게 등)에 따라 값이 크게 변하여 모든 운행 상황에 일반화되어 적용될 수 없다. 은닉 마르코프 모형(HMM, Hidden Markov model)[34], 이산 웨이블릿 변환-인공 신경망(DWT-ANN, Discrete Wavelet Transform-Artificial Neural Network)[35]을 이용한 결함 진단 방법 역시, 일부 운행 환경에 제한되어 있는 상황이다. 즉, 기존의 결함 진단 방식은 결함이 발생 여부를 파악할 수 있지만, 사용자에 의해 정의된 모든 작업 프로그램에 대해 결함이 검출되기 어려우며, 더 나아가 결함 발생 이전에 관찰될 수 있는 이상치나 고장으로 이어지는 인과적 연계성에 대해서는 분석하기 어렵다.

3.5 이상성 진단

이상치 검출은 정상 데이터로부터 미세한 차이를 보이는 이상 구간(Anomaly)을 찾기 위한 연구이다. 이 과정에서는 실제 관측한 로봇에 물리적인 이상이 존재하지 않아도, 모션 과정에서 외부의 충돌이나 흔들림에 의해 데이터 상에 큰 차이가 발생한 경우를 모두 검출하게 된다.

[36]에서는 슬라이딩 윈도우를 적용하여 데이터를 구성하고, 오토인코더(AutoEncoder) 모델을 활용하여 로봇 팔의 이상치를 검출해내는 연구를 제안하였다. 그러나 [36]의 연구에서는 실험에서 설계된 하나의

프로그램에 대한 동작하게 되므로, 사용자에 의해 정의된 다양한 프로그램을 수행하는 협동 로봇을 실시간으로 분석하여 이상치를 감지하는 데 한계가 있다. [37]에서는 연속해서 들어오는 데이터의 유사성을 기반으로 이상치를 감지하는 RBM(Restricted Boltzman Machine) 기반의 협업학습(Co-learning) 방식을 제안하였다. 구체적으로는 연속해서 들어오는 데이터로부터 비지도학습에 의해 자동 생성된 특징 벡터(Spatio-temporal Feature)를 활용하여, 일정 임계치 이상의 오차를 보이는 벡터를 검출해내는 방법이다. 비슷한 방법으로, 이상치 검출을 수행하지는 않지만 GMM(Gaussian Mixture Model)을 활용하여 작업(Task)을 식별하기 위한 시도도 존재한다[38]. 이 방법의 경우, 특정 프로그램에 대한 종속성은 존재하지 않지만, 연속적으로 들어오는 데이터의 분산이 큰 경우(즉, 다양한 형태의 프로그램을 수행하는 경우), 큰 오류를 발생시킬 수 있다. [39]에서는 로봇 팔의 센서 이상치 검출을 위한 생성적 적대 신경망(GAN, Generative Adversarial Network) 모델을 제안하였다. 특히, [39]에서는 실시간 처리를 지원하기 위해, 오프라인 학습 과정과 온라인 학습 과정을 분리하여 증분 학습(Incremental Learning)을 진행하였는데, 오프라인 학습 과정에서는 데이터의 전처리 과정 및 생성자(Generator) 및 판별자(Discriminator)를 학습하고, 온라인에서는 이를 실제 데이터에 적용하면서 모델을 업데이트하는 과정을 반복적으로 진행하도록 설계되었다. [39]의 연구 역시, 전력, 전류, 전압 등의 다양한 파라미터를 관찰하였지만, 모든 프로그램에 대해 일반화되어 적용하기에는 동일하게 한계점을 지니고 있었다. 이 외에도, 동적 환경에서 모델에 종속되어 있지 않고 이상치를 검출하기 위한 통계적인 방법[40]이나, 실시간 처리를 위한 검출 방법[41-42], 특정 파라미터(예: Angle, Energy)를 기반으로 이상치를 검출하기 위한 방법[43-44]들이 계속해서 제안되고 있다.

그러나 이와 같은 연구 역시 특정 프로그램에 대해서만 적용이 가능하거나, 모든 프로그램에 대해 적용이 된다고 해도 일시적인 이상치만 검출할 수 있어 적용성과 확장성에 제한적인 상황이다.

4. 예지 보전을 위한 모니터링 소프트웨어

4.1 예지보전 기반 요구사항

설비의 예지보전 프로세스는 데이터 수집, 상태 관리, 모델 개발, 배포 및 운영으로 이어지는 일반화된 4단계 수명주기 구조로 개념이 정립되어 있으며[45], 스마트팩토리 및 협동 로봇의 관련 연구에서도 동일한 구조가 요구사항 분류의 기준으로 활용되고 있다[45].

예지보전의 4단계 수명주기 구조를 기반으로 상태 모니터링 기술의 기능 및 비기능 요구사항을 체계화하기 위해, 운영(Operation), 관리(Management), 개발(Development), 배포(Deployment)의 네 단계로 구성하였다. 또한 각 단계의 요구사항을 실적용 가능한 상태 모니터링 기술의 사양으로 연계하기 위해 단계별 핵심 요구 속성을 정의한다.

4.2 국제표준 기반 상태 모니터링 요구사항

예지보전 수명주기(Operation, Management, Development, Deployment)에 따라 상태 모니터링 기술이 충족해야 할 핵심 요구사항을 다음 표 1과 같이 정리하였다. 각 단계의 요구사항은 설비 상태를 지속적으로 관찰하고 변화를 감지하기 위한 데이터 수집과 상태 파악(Operation), 수집된 상태 정보의 신뢰성과 일관성을 유지하기 위한 관리 기준(Management), 상태 분석 및 이상 징후 해석의 품질을 확보하기 위한 분석 체계(Development), 그리고 모니터링 결과를 운영 환경에 안정적으로 전달 및 적용하기 위한 요구사항(Deployment)을 중점적으로 고려하였다.

설비 상태 감시, 로봇 안전, 소프트웨어 품질 및 기능에 대한 다각도의 요구사항 도출을 위해, 관련 국제 표준(ISO 17359[46], ISO/TS 15066[19], ISO/IEC 25010[47], ISO 26262-6[48])을 기반으로 요구사항 속성 및 하위 요구사항을 다음 표 1과 같이 도출하였다.

표 1. 예지 보전 단계별 상태 모니터링 요구 사항[49]

예지보전 단계	요구사항 속성	요구사항 명세
Operation	Acquisition	설비 모니터링은 데이터를 시각화에 최적화된 형태로 처리해야 한다.
	Monitoring	설비 모니터링은 협동로봇의 운영 현황을 직관적으로 파악할 수 있도록 정보를 구조화하여 시각화해야 한다.
	Detection	설비 모니터링은 이상 징후 발생 시 정상 패턴 대비 편차를 사용자가 식별할 수 있도록 시각화하여 제공해야 한다.
	Safety	설비 모니터링은 안전 관련 이벤트를 신속하게 탐지하고, 경고 알림 및 시각적 강조를 통해 사용자가 상황을 즉각적으로 인지할 수 있도록 해야 한다.
	Communication	설비 모니터링은 통신 상태와 제어 신호 변화를 직관적이고 즉각적으로 시각화해야 한다.
Management	Governance	설비 모니터링은 시각화의 정확성과 일관성을 보장하기 위해, 데이터정보가 안정적이고 왜곡 없이 표현되도록 해야 한다.
	Quality	설비 모니터링은 데이터 품질 관리를 통해 화면에 표현되는 정보의 신뢰성을 확보해야 한다.
	Alignment	설비 모니터링은 데이터의 시각화 정보가 시간적으로 정확하고 일관되게 표현되도록 보장해야 한다.
	Security	설비 모니터링 화면에 노출되는 정보가 보안 규정을 충족하도록 해야 한다.

예지보전 단계	요구사항 속성	요구사항 명세
Development	Interpretability	설비 모니터링은 데이터가 화면에서 의미 있게 해석되도록 해야 한다.
	Evaluation	설비 모니터링은 화면에 표현되는 판단 결과가 운영 흐름을 정확히 반영하고 편향 없이 해석되도록 보장해야 한다.
	Robust	설비 모니터링은 운영 환경 변화나 데이터 변동이 발생해도 결과가 안정적이고 일관된 형태로 시각화되도록 해야 한다.
	Explain	설비 모니터링이 도출한 판단이나 결과를 사용자가 직관적으로 이해할 수 있도록, 근거와 맥락을 시각적 요소와 함께 명확하게 설명할 수 있어야 한다.
	Version	설비 모니터링의 설정, 사용 데이터 등 주요 구성 요소를 일관된 방식으로 관리하여, 화면에 표시되는 결과가 어떤 모델 기반인지 명확하게 식별할 수 있어야 한다.
Deployment	Interoperability	설비 모니터링은 다양한 장비·환경·플랫폼과 연동되는 과정에서도 화면의 일관성과 시각적 흐름을 유지해야 한다.
	Efficiency	설비 모니터링은 결과가 운영 환경에서 지연 없이 정확히 표현될 수 있도록 해야 한다.
	Adaptability	설비 모니터링의 변경이나 업데이트는 화면에 표시되는 정보 흐름과 운영 안정성을 해치지 않는 방식으로 안전하게 수행되어야 하며, UI는 과정 중에도 일관된 상태를 유지해야 한다.
	Resilience	설비 모니터링의 잘못된 결과나 판단이 화면에 표시되지 않도록 적절한 처리가 이루어져야 한다.
	Distribution	설비 모니터링 현장의 자원이나 조건이 달라지더라도, UI에 표시되는 모니터링 품질과 분석 결과는 안정적으로 유지되어 사용자에게 일관된 정보를 전달해야 한다.

4.3 스마트팩토리 모니터링 소프트웨어 설계

스마트팩토리 모니터링 소프트웨어의 설계 및 기능 구현을 위해 상태 모니터링 요구사항을 구체화한다. 상태 모니터링 요구사항의 속성 및 세부 요구사항에 대한 복합적 고려 및 분석을 통해 모니터링 소프트웨어에 필수적으로 요구되는 기능으로 구체화 하였으며 공통적인 속성을 지니는 기능을 종합하여 모니터링 소프트웨어 4개의 계층(Factory Overview, Live Monitoring, Fault Monitoring, Health Monitoring)을 그림 5와 같이 정의하였다. 상태 모니터링 요구사항의 기능 연계 및 구체화 결과는 표 2와 같다.

- **System UI Framework Layer:** 예지보전 요구사항을 UI 구조에 일관되게 반영하기 위한 상위 프레임워크
- **Factory Overview Layer:** 공장 단위의 운영 상태를 한눈에 파악하기 위한 상위 시각화 모듈로, 로봇 플릿(fleet, 동일 공정 또는 공장 내에서 함께 운용되는 로봇 집합) 전체의 가동 현황과 이상 징후를 통합적으로 제공
- **Live Monitoring Layer:** 개별 로봇의 동작 상태를 실시간으로 관찰하고, 운전 중 발생하는 이상 상황과 안전 상태를 즉각적으로 인지할 수 있는 시각화 모듈
- **Fault Monitoring Layer:** 로봇 시스템에서 발생하는 고장(Fault)과 그 진행 양상을 정량적으로 분석
- **Health Monitoring Layer:** 로봇 및 조인트 단위의 장기적 성능 저하와 건강 상태를 종합적으로 평가

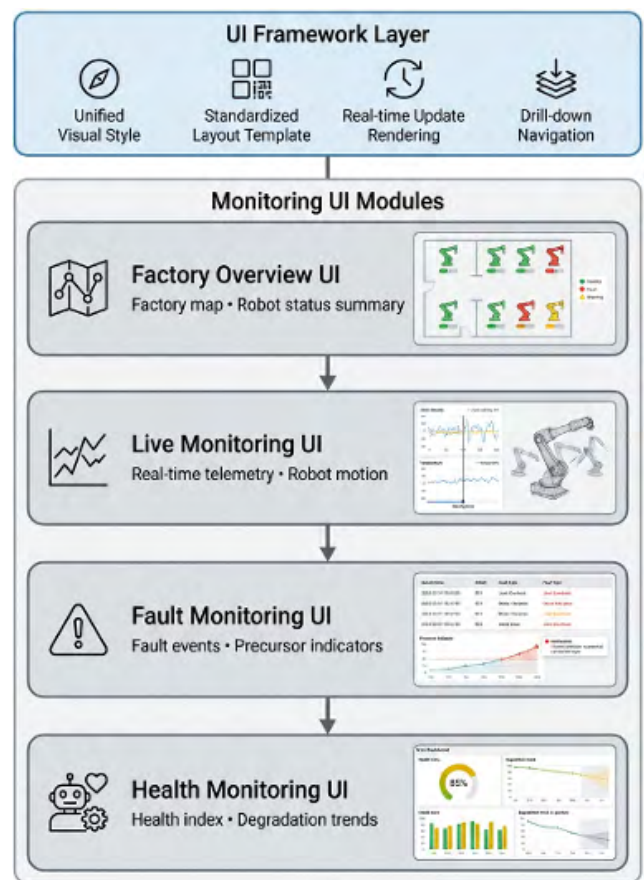


그림 5. 모니터링 소프트웨어 전체 UI 구조

표 2. 스마트팩토리 모니터링 소프트웨어의 상태 모니터링 요구사항 연계 및 구체화[49]

Layer		요구사항 구체화	관련 요구사항
System UI Framework Layer	L1-1	통합 시각 언어(UI Consistency)	Operate-Monitoring, Deploy-Interoperability
	L1-2	시간 해상도 기반 시각화	Operate-Monitoring, Operate-Acquisition
	L1-3	데이터 타입별 시각화 매핑	Operate-Monitoring, Manage-Governance, Manage-Quality
	L1-4	상태 기반 색상/아이콘 시스템	Operate-Monitoring, Operate-Detection, Operate-Safety
	L1-5	통일된 선택·필터 구조	Operate-Monitoring, Manage-Governance
	L1-6	표준화된 레이아웃 구조	Manage-Governance, Manage-Governance, Deploy-Interoperability
	L1-7	실시간 데이터 반영	Operate-Acquisition, Operate-Acquisition, Deploy-Efficiency
	L1-8	레이어 간 드릴다운(drill-down) 네비게이션	Operate-Monitoring, Operate-Detection, Deploy-Interoperability
Factory Overview Layer	L2-1	로봇 플릿 개요 대시보드 구현	Operate-Monitoring, Deploy-Interoperability
	L2-2	상태 이상 징후의 시각적 강조	Operate-Monitoring, Operate-Detection, Operate-Safety-
	L2-3	공장 레이아웃 기반 위치 맵	Operate-Monitoring, Manage-Governance, Deploy-Interoperability
	L2-4	공장 수준 집계 KPI 제공	Operate-Monitoring, Operate-Acquisition, Manage-Quality
	L2-5	로봇별 가동률·고장 시간 비교 그래프	Operate-Monitoring, Operate-Detection, Manage-Alignment
	L2-6	주기적 데이터 갱신 및 상태 표시	Operate-Acquisition, Deploy-Efficiency, Deploy-Resilience
Live Monitoring Layer	L3-1	로봇 동작의 실시간 시각화	Operate-Monitoring, Operate-Acquisition, Deploy-Efficiency
	L3-2	현재 작업 및 운전 상태 실시간 표시	Operate-Monitoring, Operate-Communication, Operate-Monitoring
	L3-3	안전-인터락 상태의 즉각 시각적 피드백	Operate-Safety, Operate-Safety, Operate-Safety, Operate-Communication
	L3-4	충돌 위험의 실시간 분석 시각화	Operate-Safety, Operate-Detection, Operate-Detection, Develop-Explain
	L3-5	Fault 발생의 즉각적 배너 알림	Operate-Detection, Operate-Safety, Deploy-Resilience, Develop-Explain
Fault Monitoring Layer	L4-1	기간 단위 Fault 통계 제공	Operate-Monitoring, Manage-Alignment, Operate-Detection
	L4-2	장기간 누적 지표제공	Manage-Quality, Manage-Governance, Deploy-Interoperability
	L4-3	Daily Stats 기반 하루 단위 세부 Fault 분석	Operate-Monitoring, Operate-Detection, Manage-Alignment
	L4-4	Fault Event 타임라인 시각화	Operate-Detection, Operate-Safety, Manage-Alignment
	L4-5	Fault 유형별 분포 및 밀도 분석 시각화	Develop-Explain, Operate-Detection, Manage-Governance
	L4-6	Precursor 기반 위험도 시각화	Operate-Detection, Develop-Explain, Develop-Robust
	L4-7	조인트별 Top Faulted Ranking	Operate-Monitoring, Develop-Explain, Manage-Quality
	L4-8	운영 영향 및 유지보수 권고사항 UI	Develop-Explain, Operate-Safety, Deploy-Resilience
Health Monitoring Layer	L5-1	기간·대상 필터 기반 Health 분석 UX	Operate-Monitoring, Deploy-Interoperability, Develop-Representation
	L5-2	디바이스-조인트 단위 Health Level 시각화	Develop-Explain, Operate-Detection, Develop-Representation
	L5-3	누적 손실 기반 저하 추세 시각화	Develop-Explain, Develop-Robust, Deploy-Efficiency
	L5-4	날짜별 Health 기록 테이블 제공	Manage-Governance, Manage-Integrity, Manage-Alignment
	L5-5	디바이스 Health 마커 기반 3D 시각화	Operate-Acquisition, Develop-Representation, Manage-Quality
	L5-6	조인트별 Raw Telemetry 드릴다운	Operate-Monitoring, Develop-Explain, Deploy-Resilience

이 중, 'Live Monitoring', 'Fault Monitoring', 'Health Monitoring' UI의 일부를 예시하면 다음 그림 6, 7, 8과 같다.

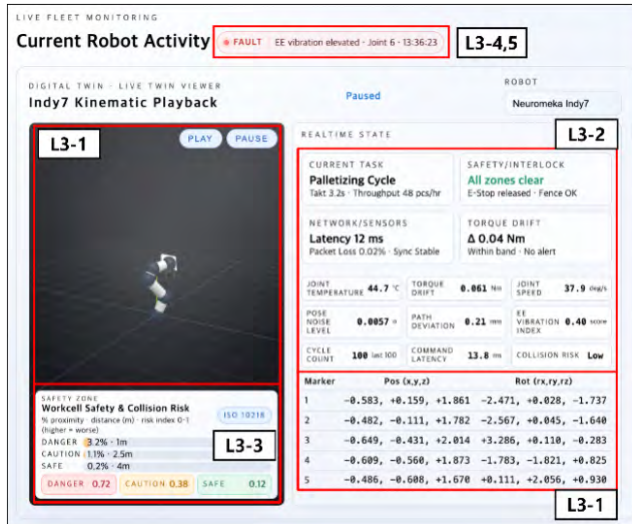


그림 6. Live Monitoring UI

- **그림 6 (L3-1 ~ 2):** 로봇 동작의 실시간 시각화(L3-1)를 위해, 로봇의 관절 움직임과 동작 궤적을 실시간으로 갱신되는 시각 요소로 표현하였다. 이와 함께 현재 작업 및 운전 상태 표시(L3-2)를 통해 수행 중인 작업 단계, 사이클 상태, 운전 모드 등의 정보를 화면 상단에 명확히 표시하여, 사용자가 로봇의 현재 상태를 즉시 파악할 수 있도록 하였다.
- **그림 6 (L3-3):** 안전 관련 요구사항을 충족하기 위해 안전 인터락 상태의 즉각적 시각적 피드백(L3-3)을 적용하였다. 인터락(interlock)은 위험 상황에서 로봇의 동작을 제한하는 안전 제어 상태를 의미하며, 본 화면에서는 인터락 및 비상정지(E-stop) 활성 여부를 색상 변화와 아이콘을 통해 즉시 확인할 수 있도록 구성하였다.
- **그림 6 (L3-4 ~ 5):** 또한 충돌 위험의 실시간 분석 시각화(L3-4)를 위해 작업자와 로봇 간 거리, 접근 속도, 위험 점수를 종합한 충돌 위험 지표를 제공하며, 위험 수준에 따라 시각적 강조 강도를 차등 적용하였다. 이상 상황이 감지될 경우에는 Fault 발생의 즉각적 배너 알림(L3-5)을 통해 화면 상단에 경고 메시지를 표시하여, 사용자가 지연 없이 상황을 인지하고 대응할 수 있도록 설계하였다

다음 그림 7은 Fault Monitoring UI를 예시한다.

- **그림 7 (L4-3 ~ 4):** Daily Stats 기반 하루 단위 세부 Fault 분석(L4-3)을 제공하여, 특정 날짜에 집중된 고장 양상이나 단기 이상 패턴을 정밀하게 분석할 수 있다. 그리고 고장의 시간적 흐름을 파악하기 위해 Fault Event 타임라인 시각화(L4-4)를 통해 고장 발생 시점을 시간 축에 따라 배열하여, 고장이 집중되는 구간이나 특정 운전 조건과의 연관성을 직관적으로 확인할 수 있다.
- **그림 7 (L4-6):** 또한 Precursor 기반 위험도 시각화(L4-6)를 포함하였다. Precursor는 고장 발생 이전에 반복적으로 관측되는 초기 이상 징후를

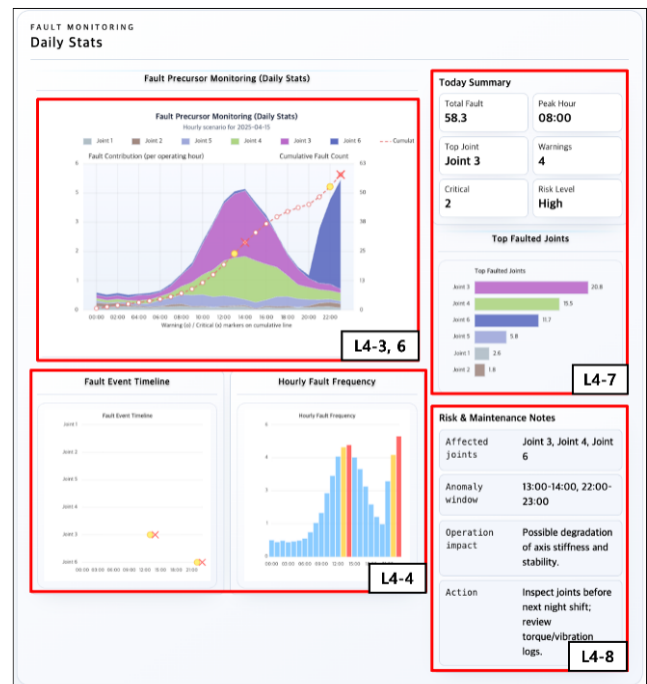


그림 7. Fault Monitoring UI

의미하며, 본 UI에서는 precursor 지표의 변화를 시각적으로 강조하여 잠재적 고장 위험을 조기에 발견할 수 있도록 지원한다.

- **그림 7 (L4-7 ~ 8):** 고장 영향의 집중도를 명확히 하기 위해 조인트별 Top Faulted Ranking(L4-7)을 제공하여, 고장이 빈번하게 발생하는 조인트를 식별할 수 있도록 하였다. 마지막으로 운영 영향 및 유지보수 권고사항 UI(L4-8)를 통해 고장이 생산성, 안전성에 미치는 영향과 함께 유지보수 권고 정보를 시각적으로 제시함으로써, 분석 결과가 실제 운영 의사결정으로 연결되도록 구성하였다

마지막으로 그림 8은 Health Monitoring UI로, 디바이스 Health 마커 기반 3D 시각화를 통해 로봇 조인트의 실제 동작 궤적을 3차원 공간에서 표현하였다. Health 분석 UX(L5-6)를 제공하여, 사용자가 특정 로봇, 조인트, 기간 조건을 선택해 분석 범위를 유연하게 조정 가능하다.

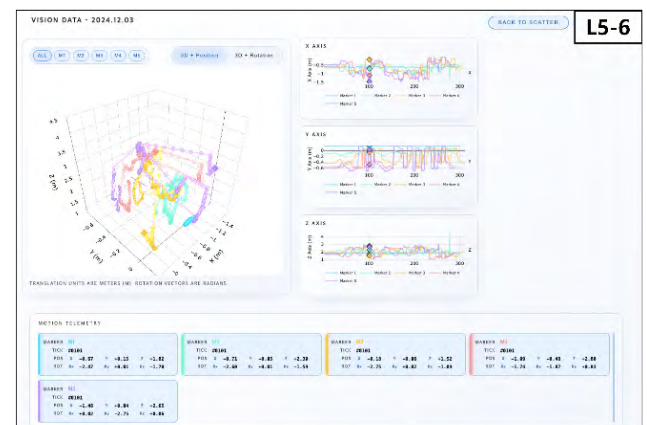


그림 8. Health Monitoring UI

5. 결론

스마트 팩토리 장비의 결함 예측, 고장 진단을 수행하는 예지 보전(Predictive Maintenance, PdM) 기술은 매우 중요하다. 그 중 협동로봇의 결함 진단은 프로그래머블한 특성, 모션의 다양성 등으로 인해 높은 성능을 내는 예지 보전 기술의 산업적 기대효과가 매우 크다. 본문에서 협동 로봇의 예지 보전기술을 소개하고, 협동 로봇의 결함 특징을 고려한 예지 보전 기술 및 모니터링 소프트웨어 설계 사례를 소개하였다. 예지 보전은 고장 예측뿐만 아니라, 나아가 장비의 노후화 및 건전성을 예측함으로써, 유지 보수 비용을 감축시키고, 공정 및 생산 제품의 품질을 보장할 수 있는 중요한 기술로 자리잡고 있다.

결론적으로 최근 딥러닝 기술의 발전으로 데이터 중심의 결함 예측은 예지 보전의 필수적인 기술이며 다양한 로봇, 작업, 모션에 일반화된 예지 보전 도구(모니터링 SW 및 테스트 프레임워크)의 체계적인 개발이 시급하다.

참고문헌

- [1] Boyes, Hugh, et al. "The industrial internet of things (IIoT): An analysis framework." *Computers in industry* 101 (2018): 1–12.
- [2] Li, Zhe, Yi Wang, and Ke-Sheng Wang. "Intelligent predictive maintenance for fault diagnosis and prognosis in machine centers: Industry 4.0 scenario." *Advances in Manufacturing*, vol. 5, no. 4, pp.377–387, 2017.
- [3] Ansari, Fazel, et al. "Autodidact: introducing the concept of mutual learning into a smart factory industry 4.0." *Proceedings of 15th International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2018)*, pp.61–68, Oct. 2018.
- [4] <https://www.digikey.com>, "Easy Automation with Omron TM Collaborative Robots"
- [5] M. Haarman, M. Mulders, and C. CVassiliadis, "Predictive maintenance 4.0–predict the unpredictable," PwC and Mainnovation, 2017.
- [6] Barlow, Richard, and Larry Hunter. "Optimum preventive maintenance policies." *Operations research* 8.1 (1960): 90–100.
- [7] Duarte, Jose A. Caldeira, Joao C. Taborda A. Craveiro, and Tomas Pedro Trigo. "Optimization of the preventive maintenance plan of a series components system." *International Journal of Pressure Vessels and Piping* 83.4 (2006): 244–248.
- [8] Wang, Jinjiang, et al. "A new paradigm of cloud-based predictive maintenance for intelligent manufacturing." *Journal of Intelligent Manufacturing*, vol. 28, no. 5 pp. 1125–1137, 2017.
- [9] Hamadache, Moussa, et al. "A comprehensive review of artificial intelligence-based approaches for rolling element bearing PHM: Shallow and deep learning." *JMST Advances* 1.1 (2019): 125–151.
- [10] <https://standards.ieee.org/ieee/1856/5451/>
- [11] Park, Ye-Seul, Dong-Yeon Yoo, and Jung-Won Lee. "Programmable Motion-Fault detection for a collaborative Robot." *IEEE Access* 9 (2021): 133123–133142.
- [12] Algburi, Riyadh Nazar Ali, and Hongli Gao. "Health assessment and fault detection system for an industrial robot using the rotary encoder signal." *Energies* 12.14 (2019): 2816.
- [13] Qiao, Guixiu, and Brian A. Weiss. "Quick health assessment for industrial robot health degradation and the supporting advanced sensing development." *Journal of manufacturing systems* 48 (2018): 51–59.
- [14] Izagirre, Unai, et al. "A methodology and experimental implementation for industrial robot health assessment via torque signature analysis." *Applied Sciences* 10.21 (2020): 7883.
- [15] Cui, Lingli, et al. "Research on remaining useful life prediction of rolling element bearings based on time-varying Kalman filter." *IEEE Transactions on Instrumentation and Measurement* 69.6 (2019): 2858–2867.
- [16] Guo, Liang, et al. "A recurrent neural network-based health indicator for remaining useful life prediction of bearings." *Neurocomputing*, vol. 240, pp. 98–109, 2017.
- [17] Xia, Min, et al. "A two-stage approach for the remaining useful life prediction of bearings using deep neural networks." *IEEE Transactions on Industrial Informatics* 15.6 (2018): 3703–3711.
- [18] Shi, Junchuan, et al. "Remaining useful life prediction of bearings using ensemble learning: The impact of diversity in base learners and features." *Journal of Computing and Information Science in Engineering* 21.2 (2021): 021004.
- [19] ISO/TS 15066:2016 Robots and robotic devices — Collaborative robots
- [20] Rosenstrauch, Martin J., and Jorg Kruger. "Safe human-robot-collaboration-introduction and experiment using ISO/TS 15066." 2017 3rd International Conference on Control, Automation and Robotics (ICCAR). IEEE, 2017.
- [21] Antonelli, Dario, and Sergey Astanin. "Qualification of a collaborative human-robot welding cell." *Procedia CIRP*, vol. 41, pp. 352–357, 2016.
- [22] Xue, Song, and Ian Howard. "Torsional vibration signal analysis as a diagnostic tool for planetary gear fault detection." *Mechanical Systems and Signal Processing*, vol. 100, pp. 706–728, 2018.
- [23] Kim, Yunhan, et al. "Phase-based time domain averaging (PTDA) for fault detection of a gearbox in an industrial robot using vibration signals." *Mechanical Systems and Signal Processing*, vol. 138, pp. 106544, 2020.

- [24] Yao, Dechen, et al. "Railway rolling bearing fault diagnosis based on multi-scale intrinsic mode function permutation entropy and extreme learning machine classifier." *Advances in Mechanical Engineering*, vol. 8, no.10, pp. 1687814016676157, 2016.
- [25] Tong, Qingbin, et al. "A fault diagnosis approach for rolling element bearings based on RSGWPT-LCD bilayer screening and extreme learning machine." *IEEE Access* 5, pp. 5515–5530, 2017.
- [26] Park, Jungho, et al. "A positive energy residual (PER) based planetary gear fault detection method under variable speed conditions." *Mechanical Systems and Signal Processing*, vol. 117, pp. 347–360, 2019.
- [27] Oh, Hyunseok, et al. "A robust and convex metric for unconstrained optimization in statistical model calibration—probability residual (PR)." *Structural and Multidisciplinary Optimization*, vol. 60, no. 3, pp. 1171–1187, 2019.
- [28] Park, Jungho, et al. "Variance of energy residual (VER): An efficient method for planetary gear fault detection under variable-speed conditions." *Journal of sound and vibration*, vol. 453, pp. 253–267, 2019.
- [29] Yoo, Minji, et al. "A resilience measure formulation that considers sensor faults." *Reliability Engineering & System Safety*, 2019.
- [30] Han, Te, et al. "Deep transfer network with joint distribution adaptation: A new intelligent fault diagnosis framework for industry application." *ISA transactions*, vol. 97, pp. 269–281, 2020.
- [31] Mbo'o, Christelle Plantsop, and Kay Hameyer. "Fault diagnosis of bearing damage by means of the linear discriminant analysis of stator current features from the frequency selection." *IEEE Transactions on Industry Applications*, vol. 52, no. 5, pp. 3861–3868, 2016.
- [32] Zhang, Xiaoyuan, Yitao Liang, and Jianzhong Zhou. "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM." *Measurement*, vol. 69, pp. 164–179, 2015.
- [33] Xie, Yuan, and Tao Zhang. "Fault diagnosis for rotating machinery based on convolutional neural network and empirical mode decomposition." *Shock and Vibration*, 2017.
- [34] Jaber, Alaa Abdulhady, and Robert Bicker. "Fault diagnosis of industrial robot gears based on discrete wavelet transform and artificial neural network." *Insight-Non-Destructive Testing and Condition Monitoring*, vol. 58, no. 4, pp. 179–186, 2016.
- [35] Cheng, Fangzhou, et al. "High-Accuracy Unsupervised Fault Detection of Industrial Robots Using Current Signal Analysis." *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, 2019.
- [36] Chen, Tingting, et al. "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder." *IEEE Access* 8 (2020): 47072–47081.
- [37] Munawar, Asim, Phongtharin Vinayavekhin, and Giovanni De Magistris. "Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space." *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.
- [38] Kanazawa, Akira, Jun Kinugawa, and Kazuhiro Kosuge. "Incremental Learning of Spatial-Temporal Features in Human Motion Patterns with Mixture Model for Planning Motion of a Collaborative Robot in Assembly Lines." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
- [39] Lu, Haodong, et al. "GAN-based Data Augmentation Strategy for Sensor Anomaly Detection in Industrial Robots." *IEEE Sensors Journal* (2021).
- [40] Hornung, Rachel, et al. "Model-free robot anomaly detection." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014.
- [41] Solch, Maximilian, et al. "Variational inference for on-line anomaly detection in high-dimensional time series." *arXiv preprint arXiv:1602.07109* (2016).
- [42] Khalastchi, Eliahu, et al. "Online data-driven anomaly detection in autonomous robots." *Knowledge and Information Systems* 43.3 (2015): 657–688.
- [43] Kriegel, Hans-Peter, Matthias Schubert, and Arthur Zimek. "Angle-based outlier detection in high-dimensional data." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008.
- [44] Pereira, Joao, and Margarida Silveira. "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention." *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018.
- [45] Jay Lee, et al., "Introduction to cyber manufacturing," *Manufacturing Letters*, vol. 8, pp. 11–15, 2016.
- [46] ISO 17359, Condition monitoring and diagnostics of machines – General guidelines, International Organization for Standardization, 2018.
- [47] ISO/IEC 25010, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model, International Organization for Standardization, 2023.
- [48] ISO 26262-6, Road vehicles – Functional safety – Part 6: Product development at the software level, International Organization for Standardization, 2018.
- [49] 황세현, 최민서, 김진세, 이정원, "스마트팩토리 예지보전을 위한 모니터링 소프트웨어 설계", KCSE 2026.



신진연구자 소개 I

DGIST 김기섭 교수



• 김기섭 박사

조교수

DGIST EECS

BASE (Bi-directional AI Software Engineering)

kisub.kim@dgist.ac.kr

<https://base.dgist.ac.kr>

주요 약력

- 2025.08~Present DGIST Electrical Engineering and Computer Science, Assistant Professor
- 2023.10~2025.06 Huawei Hong Kong Research Center, Senior Software Engineer
- 2021.12~2023.07 Singapore Management University, Research Scientist
- 2021.08~2021.10 University of Luxembourg, Research Associate
- 2018.08~2021.07 University of Luxembourg, PhD Program (Advisor: Tegawende F. Bissyandé)
- 2016.03~2018.02 Chungbuk National University (Advisor: Younghoon Seo)
- 2015.09~2016.01 Kyunghee University Medical Center, OCS Developer
- 2007.03~2015.02 Chungbuk National University Computer Engineering

대표논문

1. Xin Zhou, **Kisub Kim****, Ting Zhang, Martin Weyssow, Luís F Gomes, Guang Yang, Kui Liu, Xin Xia, and David Lo, "SE-Jury: An LLM-as-Ensemble-Judge Metric for Narrowing the Gap with Human Evaluation in SE," IEEE/ACM International Conference on Automated Software Engineering (ASE), 2025.
2. **Kisub Kim**, Jonghoon Kim, Byeongjo Park, Dongsun Kim, Chun Yong Chong, Yuan Wang, Tiezhu Sun, Xunzhu Tang, Jacques Klein, and Tegawendé F Bissyandé, "DataRecipe — How to Cook the Data for CodeLLM?" the IEEE/ACM International Conference on Automated Software Engineering (ASE), 2024.
3. **Kisub Kim**, Xin Zhou, Dongsun Kim, Jaekwon Lee, Tegawendé F Bissyandé, Jacques Klein, Kui Liu, and David Lo, "How are We Detecting Inconsistent Method Names? Leveraging Code Reviews for Building a Reliable Benchmark," ACM Transactions on Software Engineering and Methodology (TOSEM), 2024.
4. Xin Zhou, **Kisub Kim****, Bowen Xu, Donggyun Han, and David Lo, "Out of Sight, Out of Mind: Better Automatic Vulnerability Repair by Broadening Input Ranges and Sources," the IEEE/ACM International

Conference on Software Engineering (ICSE), 2024.

5. **Kisub Kim**, Sankalp Ghatpande, Dongsun Kim, Xin Zhou, Kui Liu, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon, "Big Code Search: a Bibliography," ACM Computing Surveys (CSUR), 2023.

김기섭 교수님은 2025년부터 DGIST에서 근무 중입니다. 주요 연구분야는 AI for Software Engineering 으로, 2021년에 University of Luxembourg에서 박사학위를 받으셨습니다.

박사학위 주제는 "Steps Towards Semantic Code Search"입니다. 대학에 교수직으로 돌아오시기 전에는 Huawei HKRC의 소프트웨어공학 팀에서 근무하며, AI Training에 관련한 업무를 담당하셨습니다. 현재는 AI 모델을 학습 및 튜닝 하는 연구와 더불어 뉴런 단위로 분석하고, 설계하여 흥미로운 결과들을 가지고 여러 Task에 적용하고, 발전시키는 연구를 하고 계십니다. 김기섭 교수님에 대한 보다 자세한 약력은 말미를 참조하시기 바랍니다. 아래 문답에서 "편"은 편집자를 "김"은 김기섭 교수님을 지칭합니다.

편: Code LLM은 실제로 코딩을 이해하고 있는 걸까요, 아니면 그럴듯하게 기억하고 있는 걸까요?

김: 솔직히 말씀드리면, 지금의 Code LLM은 이해와 기억 사이 어딘가에 있다고 생각합니다. 완전히 이해한다고 말하기도 어렵지만, 단순히 기존 코드를 복사하는 시스템이라고 보기에 이미 상당히 복잡한 능력을 보여주고 있습니다.

예를 들어 Code LLM은 변수명을 바꾸거나 코드 구조를 조금 변형해도 문제를 해결할 때가 있습니다. 처음 보는 라이브러리나 낯선 요구사항에도 어느 정도 대응합니다. 이런 모습을 보면 단순 암기만으로 설명하기는 어렵습니다. 반면 유명한 알고리즘이나 널리 공개된 벤치마크에서는 학습 데이터에서 본 패턴을 다시 깨내는 것만으로도 굉장히 지능적으로 보일 수 있습니다. 시험 문제와 해설을 모두 외운 학생이 높은 점수를 받았다고 해서, 그 학생이 개념을 완전히 이해했다고 단정할 수 없는 것과 비슷합니다.

문제는 우리가 모델의 정답만 본다는 데 있습니다. 답을 맞히면 "이해했다"고 생각하기 쉽지만, 사실은 우연히 익숙한 패턴을 만났을 수도 있습니다. 그래서 저는 모델이 정답을 맞췄지만 보는 것보다, 문제의 표현을 바꾸고, 변수명을 바꾸고, 불필요한 정보를 넣거나 일부 문맥을 제거했을 때에도 일관되게 해결하는지를 봐야 한다고 생각합니다. 비슷한 문제를 조금만 비틀었는데 성능이 급격히 떨어진다면, 이해보다는 기억에 의존했을 가능성이 큼니다.

다만 여기서 더 재미있는 질문은 “인간 개발자는 정말 이해만으로 코딩하는가?”입니다. 사람도 Stack Overflow의 코드를 가져오고, 익숙한 디자인 패턴을 재사용하며, 이전 프로젝트의 경험을 떠올립니다. 인간 역시 상당 부분 기억을 활용합니다. 차이가 있다면 사람은 코드가 실패했을 때 실행해 보고, 디버깅하고, 가정을 수정하면서 자신의 이해를 갱신할 수 있다는 점입니다.

따라서 Code LLM이 코딩을 이해하는지를 추상적으로 논쟁하는 것보다, 낮은 상황에서도 자신의 답을 검증하고, 실패 원인을 찾아 수정할 수 있는가를 묻는 것이 더 생산적이라고 생각합니다. 결국 이해는 한 번에 정답을 내는 능력보다, 틀렸을 때 왜 틀렸는지를 알아내는 능력에서 더 잘 드러나기 때문입니다.

편: 시가 코드를 더 많이 만들수록, 소프트웨어는 오히려 더 신뢰하기 어려워지는 것은 아닐까요?

김: 그럴 가능성이 충분히 있다고 생각합니다. AI는 코드를 작성하는 비용을 크게 낮췄지만, 아직 그 코드를 신뢰하는 비용까지 낮춰 준 것은 아니기 때문입니다.

예전에는 개발자가 코드 100줄을 작성하고 그 100줄을 검토했다면, 앞으로는 시가 몇 초 만에 1,000줄을 만들고 개발자가 그것을 검토해야 할 수도 있습니다. 코드 생산성은 열 배가 되었는데, 사람의 검토 능력은 열 배가 되지 않습니다. 자칫하면 시가 개발자의 업무를 줄이는 것이 아니라, 검토해야 할 코드의 양만 늘릴 수 있습니다.

더 큰 문제는 시가 만든 코드가 상당히 그럴듯하다는 점입니다. 문법적으로 자연스럽고, 주석도 있고, 테스트도 일부 통과하기 때문에 얼핏 보면 잘 작성된 코드처럼 보입니다. 하지만 존재하지 않는 API를 호출하거나, 예외 상황을 놓치거나, 보안상 위험한 기본 설정을 사용할 수 있습니다. 완전히 엉뚱한 코드는 오히려 발견하기 쉽지만, 거의 맞는 코드는 더 위험할 수 있습니다.

시가 생성하는 코드의 양이 늘어나면 소프트웨어 공급망도 더 복잡해질 수 있습니다. 모델이 어떤 라이브러리를 왜 선택했는지, 생성한 코드가 어느 버전에 의존하는지, 라이선스나 보안 문제는 없는지, 배포 환경에서도 동일하게 동작하는지를 확인해야 합니다. 코드를 만드는 행위는 자동화되지만, 그 코드가 시스템 전체에 미치는 영향을 추적하는 일은 오히려 더 어려워질 수 있습니다.

그렇다고 AI 코드 생성을 부정적으로만 볼 필요는 없습니다. 핵심은 생성 자동화와 검증 자동화가 함께 발전해야 한다는 것입니다. 시가 코드를 열 배 빠르게 생성한다면, 테스트 생성, 정적 분석, 취약점 검사, 의존성 검증, 변경 영향 분석도 그에 맞춰 자동화되어야 합니다. 생성하는 시와 검증하는 시를 분리하거나, 여러 에이전트가 서로의 결과를 비판하도록 만드는 것도 하나의 접근이 될 수 있습니다. 물론 시가 시를 검증한다고 해서 자동으로 신뢰할 수 있는 것은 아니므로, 중요한 의사결정에는 명시적인 검증 기준과 사람의 개입이 필요합니다.

저는 AI 시대의 소프트웨어 위기를 “코드가 부족한 문제”가 아니라 검증되지 않은 코드가 너무 많아지는 문제라고 봅니다. 앞으로 중요한 경쟁력은 누가 더 많은 코드를 생성하느냐보다, 누가 생성된 코드를 더 안전하게 선별하고 통합하며 운영할 수 있느냐에서 나올 가능성이 큼니다.

편: 앞으로 개발자의 핵심 역량은 코드를 잘 작성하는 능력일까요, 시가 작성한 코드를 의심하는 능력일까요?

김: 둘 중 하나를 고르라면, 앞으로는 시가 작성한 코드를 제대로 의심하는 능력이 더 중요해질 수 있다고 생각합니다. 다만 코드를 잘 모르는 사람이 좋은 의심을 할 수는 없습니다. 결국 코드를 작성하는 능력이 사라지는 것이 아니라, 그 능력의 쓰임이 달라지는 것입니다.

예전에는 개발자의 실력이 “얼마나 빠르고 정확하게 코드를 작성하는가”로 많이 평가되었습니다. 앞으로는 시가 초안을 빠르게 만들어 줄 수 있기 때문에, 개발자는 그 코드가 요구사항을 정확히 반영했는지, 숨겨진 가정은 무엇인지, 어떤 상황에서 실패하는지 판단해야 합니다. 직접 문장을 쓰는 능력보다 편집자의 판단력이 중요해지는 것과 비슷합니다.

여기서 말하는 ‘의심’은 무조건 AI의 답을 불신하라는 뜻은 아닙니다. 좋은 개발자는 다음과 같은 질문을 할 수 있어야 합니다.

“왜 이 라이브러리를 선택했지?”

“동시에 여러 요청이 들어오면 어떻게 되지?”

“입력값이 비어 있거나 예상보다 크면?”

“테스트는 통과했지만 요구사항을 잘못 이해한 것은 아닐까?”

“이 코드를 1년 뒤에 다른 개발자가 수정할 수 있을까?”

이런 질문은 단순한 문법 지식만으로는 나오지 않습니다. 시스템 구조, 데이터 흐름, 성능, 보안, 유지보수, 사용자 요구를 종합적으로 이해해야 합니다. 그래서 역설적으로 시가 코드를 더 많이 작성할수록 개발자에게 더 깊은 기초 지식이 필요할 수 있습니다. 시가 계산기를 제공했다고 해서 수학적 사고가 필요 없어지지 않는 것과 같습니다. 계산은 빨라졌지만, 무엇을 계산해야 하는지 판단하는 책임은 여전히 사람에게 남아 있습니다.

개발자의 역할도 ‘코드를 생산하는 사람’에서 ‘소프트웨어에 대한 의사결정을 내리는 사람’으로 이동할 가능성이 큼니다. 문제를 정의하고, 아키텍처를 설계하며, 시에 적절한 문맥을 제공하고, 결과를 검증하고, 실패에 대한 책임을 관리하는 역할입니다. 특히 여러 AI Agent가 개발 과정에 참여하게 되면 개발자는 모든 코드를 직접 작성하기보다, 에이전트의 역할과 권한을 설계하고 결과를 조율하는 일종의 오케스트레이터가 될 수 있습니다.

학생들에게는 AI를 쓰지 말라고 할 것이 아니라, AI의 결과를 그대로 믿지 않는 훈련을 시켜야 한다고 생각합니다. 시가 만든 코드를 직접 실행하고, 테스트를 추가하고, 일부 조건을 바꿔 보고, 실패 원인을 설명해 보는 연습이 중요합니다.

결국 미래의 좋은 개발자는 AI보다 코드를 빨리 타이핑하는 사람이 아닐 것입니다. 시가 놓친 것을 발견하고, 시가 자신 있게 틀렸을 때 멈춰 세울 수 있는 사람이 좋은 개발자가 될 것이라고 생각합니다.

편: Code LLM의 성능이 계속 높아지고 있는데요. 벤치마크 점수가 높으면 실제로도 좋은 개발자라고 볼 수 있을까요?

김: 저는 벤치마크 점수가 높다는 것을 시험을 잘 본 것이라고는 말할 수 있지만,

곧바로 좋은 개발자와 같은 능력을 갖췄다고 말하기는 어렵다고 생각합니다.

현재 Code LLM 벤치마크에서는 주어진 문제에 대해 코드를 생성하고, 테스트를 통과했는지를 중심으로 성능을 평가하는 경우가 많습니다. 이 방식은 명확하고 비교하기 쉽다는 장점이 있습니다. 하지만 실제 소프트웨어 개발은 문제 설명을 읽고 정답 코드를 한 번에 작성하는 시험과는 상당히 다릅니다.

현실에서는 요구사항 자체가 불명확한 경우가 많습니다. 어떤 기능을 수정하면 다른 기능이 깨질 수 있고, 코드뿐만 아니라 이슈 보고서, 문서, 테스트, 변경 이력, 빌드 설정까지 함께 이해해야 합니다. 때로는 사용자가 요청한 대로 구현하는 것이 오히려 잘못된 선택일 수도 있습니다. 요구사항의 모순을 발견하고 다시 질문하는 것도 개발자의 중요한 능력입니다.

그런데 현재 벤치마크에서는 모델이 문제를 잘못 이해했더라도 우연히 테스트만 통과하면 정답으로 처리될 수 있습니다. 반대로 기능적으로는 올바른 패치를 만들었지만 벤치마크에 포함된 테스트와 조금 다르게 구현하면 실패로 평가될 수도 있습니다. 테스트 통과 여부는 중요하지만, 그것만으로 코드의 품질과 모델의 능력을 모두 설명할 수는 없습니다.

또 하나의 문제는 모델이 평가 문제를 이전에 보았을 가능성입니다. 공개된 저장소와 이슈를 이용해 벤치마크를 만들면, 그 내용이 이미 모델의 학습 데이터에 포함되어 있을 수 있습니다. 이런 경우 점수가 높은 이유가 문제 해결 능력 때문인지, 과거에 본 내용을 기억했기 때문인지 구분하기 어렵습니다. 시험 전에 문제를 본 학생과 처음 보는 학생을 같은 기준으로 비교하는 셈입니다.

그래서 저는 앞으로의 평가는 한 번의 정답보다 문제를 해결하는 과정을 더 많이 봐야 한다고 생각합니다. 어떤 파일을 확인했는지, 어떤 가설을 세웠는지, 실패한 테스트를 어떻게 해석했는지, 첫 번째 패치가 틀렸을 때 어떻게 수정했는지를 평가해야 합니다. 같은 저장소가 시간이 지나며 변화할 때 모델이 지속적으로 대응할 수 있는지도 중요합니다.

결국 벤치마크는 모델 능력의 일부를 측정하는 체온계와 비슷합니다. 체온은 건강 상태를 파악하는 데 유용하지만, 체온 하나만으로 그 사람이 건강하다고 단정할 수는 없습니다. Code LLM도 단일 점수로 줄을 세우기보다, 정확성, 강건성, 보안성, 유지보수성, 비용을 함께 평가해야 합니다.

좋은 Code LLM은 문제를 많이 맞히는 모델을 넘어, 자신이 무엇을 모르는지 알고 필요할 때 확인하는 모델이어야 한다고 생각합니다.

편: 범용 Code LLM의 성능이 계속 좋아진다면, 기업마다 별도의 AI 모델을 만들 필요가 있을까요?

김: 범용 모델이 아무리 좋아져도 기업 맞춤형 AI의 필요성은 사라지지 않을 가능성이 큼니다. 소프트웨어 개발에서 어려운 부분은 프로그래밍 언어 자체보다, 그 회사만의 문맥을 이해하는 것이기 때문입니다.

예를 들어 어떤 함수가 기술적으로는 잘 작성되었더라도, 회사 내부의 아키텍처 규칙을 위반하거나 승인되지 않은 라이브러리를 사용하면 실제로는 쓸 수 없습니다. 같은 기능이라도 기업마다 사용하는 프레임워크, 코딩 규칙, 보안 정책, 배포 환경이 다릅니다. 오래된 시스템에서는 문서에 없는 관행이나 예외 처리

방식이 코드 속에만 남아 있기도 합니다.

범용 모델은 Python이나 Java에 대해서는 많이 알고 있지만, 특정 기업에서 왜 이 모듈을 수정하면 안 되는지, 왜 이상해 보이는 코드가 10년째 유지되고 있는지까지는 알기 어렵습니다. 개발 현장에서는 이런 지역적인 지식이 오히려 더 중요할 때가 많습니다. 코드만 보면 불필요해 보이는 조건문이 과거 장애를 막기 위해 추가된 것일 수도 있기 때문입니다.

그렇다고 기업마다 거대한 모델을 처음부터 학습해야 한다는 의미는 아닙니다. 저는 기업 맞춤형 Code LLM을 단순히 파인튜닝된 하나의 모델로 보지 않습니다. 범용 모델 위에 기업의 코드, 문서, 이슈, 테스트, 개발 규칙을 검색해 제공하고, 정적 분석이나 빌드 시스템과 연결하며, 기업별 검증 기준을 적용하는 전체 시스템으로 봐야 합니다.

특히 기업 내부 데이터는 많다고 해서 모두 좋은 학습 데이터가 되는 것은 아닙니다. 오래된 코드, 복사된 코드, 실패한 패치, 일관되지 않은 규칙도 함께 들어 있습니다. 이를 무작정 학습시키면 모델이 그 기업의 장점뿐 아니라 기술 부채까지 충실하게 배울 수 있습니다. 기업 맞춤형에서는 무엇을 학습시킬지보다, 무엇을 학습시키지 않을지를 결정하는 일이 더 어려울 수도 있습니다.

또한 모든 기업이 같은 수준의 자동화를 원하는 것도 아닙니다. 어떤 기업은 코드 추천만 허용할 수 있고, 어떤 기업은 테스트 실행까지 맡길 수 있으며, 일부 환경에서는 배포까지 자동화할 수 있습니다. 따라서 모델의 성능뿐만 아니라 권한, 보안, 책임 범위를 기업 상황에 맞게 설계해야 합니다.

결국 범용 모델은 똑똑한 신입 개발자와 비슷하다고 생각합니다. 프로그래밍 지식은 풍부하지만, 회사의 시스템과 역사, 업무 방식을 아직 모릅니다. 신입 개발자가 조직에서 제대로 일하려면 온보딩이 필요하듯이, 범용 Code LLM에도 기업별 온보딩이 필요합니다.

앞으로 기업 간 AI 경쟁력의 차이는 누가 더 큰 모델을 보유했느냐보다, 자신의 소프트웨어 지식과 개발 경험을 얼마나 잘 정리해 AI가 활용하도록 만들었느냐에서 나타날 것이라고 생각합니다.



신진연구자 소개 II

충남대학교 김형석 교수



• 김형석 박사

조교수
충남대학교 컴퓨터인공지능학부
SWSEC 연구실
hyungseok.kim@cnu.ac.kr
<https://sites.google.com/view/hyungseok-kim/>

주요 약력

- 2025.09~현재 충남대학교 컴퓨터인공지능학부 조교수
- 2016.01~2025.08 국가보안기술연구소 선임연구원
- 2011.12~2015.12 국가보안기술연구소 연구원
- 2024.02 KAIST 정보보호대학원 박사 (지도교수: 차상길)
- 2012.02 KAIST 전산학과 석사 (지도교수: 김대영)
- 2010.02 숭실대학교 컴퓨터학부 학사

대표논문

- [1] Hyungseok Kim, Soomin Kim, and Sang Kil Cha, Towards Sound Reassembly of Modern x86-64 Binaries, In Proceedings of ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 1317-1333, April 2025.
- [2] Hyungseok Kim, Soomin Kim, Jungwoo Lee, and Sang Kil Cha, "AsFuzzer: Differential Testing of Assemblers with Error-Driven Grammar Inference," In Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pages 1099-1111, September 2024. (Distinguished Paper Award)
- [3] Hyungseok Kim, Soomin Kim, Junoh Lee, Kangkook Jee, and Sang Kil Cha, "Reassembly is Hard: A Reflection on Challenges and Strategies," In Proceedings of USENIX Security Symposium (USENIX Security), pages 1469-1486, August 2023.
- [4] Hyungseok Kim, Junoh Lee, Soomin Kim, Seungil Jung, and Sang Kil Cha, "How'd Security Benefit Reverse Engineers? The Implication of Intel CET on Function Identification," In Proceedings of the International Conference on Dependable Systems Networks (DSN), pages 559-566, June 2022.

김형석 교수님은 2025년부터 충남대학교에서 근무 중입니다. 주요 연구분야는 소프트웨어 보안으로, 2024년에 KAIST에서 박사학위를 받으셨습니다.

김형석 교수님의 박사학위 논문 주제는 "Towards Reliable Reassembly with Hardware-assisted Stateless CF"로, 소스 코드가 제공되지 않는 바이너리 프로그램의 안전하고 신뢰할 수 있는 수정 기술을 연구하였습니다. 교수로 부임하기 전에는 국가보안기술연구소에서 근무하며 국가용 보안 소프트웨어

개발과 소프트웨어 시스템 취약점 분석 업무를 수행하였습니다. 이러한 연구 및 실무 경험을 바탕으로 현재는 소스 코드가 없는 소프트웨어의 취약점을 분석하고 이를 패치하기 위한 다양한 연구를 수행하고 있습니다. 최근에는 바이너리 프로그램의 취약점을 효과적으로 탐지하기 위해 기존의 정적-동적 프로그램 분석 기술과 대규모 언어 모델(LLM)을 접목하는 연구에 관심을 두고 있습니다. 김형석 교수님에 대한 자세한 약력은 글 말미에서 확인하실 수 있습니다. 아래 문단에서는 '편'은 편집자를, '김'은 김형석 교수님을 지칭합니다.

편: Software Engineering 연구를 수행하시면서, 교수님께서 생각하시는 "좋은 SE 연구"의 기준은 무엇인지 궁금합니다. 특히 최근 AI 기반 SE 연구가 빠르게 확산되는 상황에서, 변하지 않아야 할 핵심 가치는 뭐라고 생각하세요??

김: 저는 리버싱, 퍼징, 정적 바이너리 계측, 취약점 분석처럼 SE와 보안이 맞닿아 있는 부분에서 주로 연구를 수행해 왔습니다. 그 관점에서 제가 생각하는 좋은 SE 연구는 실제 소프트웨어에서 발생 가능한 문제를 잘 정의하고, 현실에서 적용 가능한 해결책을 제시하는 연구라고 생각합니다.

최근 AI 기반 SE 연구가 확산되면서 코드 생성, 버그 탐지, 테스트 생성, 자동 패치 같은 분야에서 흥미로운 결과들이 많이 나오고 있습니다. 이 중에 AI가 활용되더라도 변하지 않아야 할 핵심 가치를 하나 꼽는다면, 저는 신뢰성이라고 생각합니다.

이제는 AI가 코드를 자동으로 생성하면서 개발 주기는 짧아지고 소프트웨어의 복잡도는 더 증가하고 있습니다. 개발자가 세부적인 내용을 완전히 파악하기는 점점 어려워지고 있습니다. 소프트웨어의 신뢰성은 AI가 생성한 코드뿐만 아니라 그 코드가 의존하는 라이브러리, 빌드 환경, 테스트 과정, 배포 파이프라인, 패치 과정까지 모두 신뢰의 대상이 됩니다.

이 점에서 Ken Thompson의 Reflections on Trusting Trust는 여전히 중요한 메시지를 준다고 생각합니다. 이 발표에서는 우리가 소프트웨어를 신뢰한다고 할 때 그 신뢰의 근거가 어디에 있는지를 묻고 있습니다.

AI 기반 SE 시대에서도 이 질문에 대한 고민이 필요하다고 생각합니다. 단순히 모델이 코드를 잘 생성하거나 정답을 잘 맞히는 데서 개발이 끝나게 아니라, 어떤 코드가 생성되었고, 어떤 구성요소에 의존하고, 어떤 과정으로 빌드되고 배포되며, 어디에서 신뢰가 깨질 수 있는지를 고민해 봐야 할 거 같습니다. 그런 의미에서 AI 시대에도 변하지 않아야 할 SE 연구의 핵심 가치는 신뢰성이라고 생각합니다.

편: 교수님의 연구는 바이너리 분석, 취약점 탐지, 실행 환경 보안 등 비교적 low-level 영역에 집중되어 있는것으로 보이는데요, 혹시 현재 AI 모델들 중에 취약점 탐지와 같이 보안에 최적화 되어있는, 가장 최전선에 있는 모델들을 뛰어넘기위해 오픈소스를 학습시키거나 튜닝시켜 본 연구가 있나요? 어떤 방법이 지금까지 가장 좋은가요?

김: 제가 주로 연구해 온 영역은 소스코드 분석보다는 바이너리 분석입니다. 최신 보안 특화 모델을 뛰어넘기 위해 모델을 직접 학습시키거나 튜닝해 본 경험은 없지만, 바이너리 분석 관점에서 AI 모델을 취약점 탐지에 활용하는 방법에 대해 말씀드리겠습니다.

최근 LLM을 보안 분야에 활용하려는 연구는 매우 활발합니다. 보안 특화 데이터로 오픈소스 모델을 파인튜닝하거나, 취약점 탐지, 퍼징, 프로그램 수리, 악성코드 분석 등에 LLM을 적용하려는 시도들이 많이 나오고 있습니다.

하지만, 바이너리 취약점 탐지에서는 시가 처음부터 끝까지 단독으로 해결하기 어려운 부분이 많습니다. 바이너리 코드는 사람이 작성한 소스코드와 달리 함수 이름, 변수 이름, 자료형, 주석 같은 힌트가 대부분 사라져 있습니다. 또한 자연어와 비교했을 때, 명령어 토큰들 사이의 의미 관계가 표면적으로 잘 드러나지 않습니다. 명령어 하나의 의미도 주변 토큰만으로 결정되기보다는 레지스터 상태, 메모리 상태, 제어 흐름, 이전 실행 경로에 강하게 의존합니다. 그래서 단순히 명령어 시퀀스만 보고 프로그램의 구조나 의도를 정확히 파악하기는 어렵습니다.

현재는 바이너리 분석 도구가 만든 결과를 바탕으로 시가 의미 해석이나 취약점 판단을 보조하는 방식이 가장 효과적이라고 생각합니다. 예를 들어 디어셈블러, 디컴파일러, 정적 분석기가 복원한 함수 경계, 제어 흐름 그래프 같은 정보를 LLM이 활용한다면 단순한 명령어 패턴보다 더 의미 있는 판단을 할 수 있습니다.

최근 연구들도 이런 방향으로 조금씩 이동하고 있다고 봅니다. 초기에는 딥러닝 기반으로 함수 경계 식별이나 제어흐름 정확도를 높이려는 연구들이 있었고, 최근에는 디컴파일 결과를 개선하거나 분석 도구의 결과를 LLM과 결합해 취약점 탐지를 보조하려는 연구들이 나오고 있습니다. 다만 분석기마다 목적과 특성이 다르고, 어떤 분석기는 빠르지만 부정확할 수 있고, 어떤 분석기는 정밀하지만 확장성이 떨어질 수 있습니다. 이런 다양한 분석기의 결과를 AI 모델이 어떻게 이해하고 활용하게 만들 것인지는 아직 충분히 정리된 문제는 아니라고 생각합니다.

그런 점에서 바이너리 분석과 AI의 결합은 아직 흥미로운 연구 기회가 많은 분야입니다. 저도 기존 정적 분석기의 결과 위에서 AI를 활용하는 방향은 앞으로 충분히 가능성이 있다고 보고 관심 있게 지켜보고 있습니다.

편: 과거 연구소에서의 실무 경험이 현재 연구 주제나 문제 정의 방식에 어떤 영향을 미쳤는지 궁금합니다. 특히 현장에서 느낀 한계나 문제의식이 연구로 이어진 사례가 있으신가요?

김: 국가보안기술연구소에서의 경험은 제 연구 방향에 상당히 큰 영향을 주었습니다. 연구소에 입소할 당시 저는 주로 소프트웨어 개발 경험만 가지고 있었고, 개발 부서에서 보안 소프트웨어 개발 업무를 맡았습니다.

그전까지는 소프트웨어를 잘 설계하고, 새로운 기술을 빠르게 습득하고, 능숙하게 구현하는 것이 소프트웨어 개발에 있어서 가장 중요한 부분이라고 생각했습니다. 그런데 제가 작성한 코드를 다른 부서의 연구원들이 검증하는 과정을 보면서, 소프트웨어 테스팅과 검증 절차의 중요성을 새롭게 깨닫게 되었습니다. 그리고 코드의 오류가 단순한 기능 결함에 그치는 것이 아니라, 실제 사이버 공격으로 이어질 수 있다는 점도 배울 수 있었습니다.

그 경험은 저에게 중요한 전환점이 되었습니다. 개발자 입장에서 버그는 “고쳐야 할 귀찮은 숙제” 정도로 생각했지만, 보안 연구자 관점에서는 그 버그가 공격자가 시스템을 장악하거나 정보를 유출할 수 있는 “중요한 단서”가 된다는 점이 흥미로웠습니다. 같은 문제라도 개발자와 공격자가 바라보는 관점이 크게 다르다는 것을 배웠고, 그 이후로 개발자에서 보안 연구자로 관심과 정체성이 옮겨갔습니다.

이후 연구소에서 취약점 분석 업무를 맡게 되면서 소스 코드가 없는 프로그램을 분석했습니다. 그 과정에서 바이너리 분석은 상당 부분 수작업과 분석가의 경험에 의존한다는 점을 느꼈고, 자연스럽게 분석과 검증을 어떻게 자동화할 수 있을지에 관심을 갖게 되었습니다. 이러한 문제의식이 이후 바이너리 분석 도구를 설계하고, 자동으로 테스트를 수행하며, 프로그램을 수정하는 기술을 연구하는 방향으로 이어졌습니다.

그래서 국가보안기술연구소에서의 경험은 제게 SE와 보안은 분리된 문제가 아니라 서로 깊이 연결되어 있다는 점을 인식하게 해 준 계기였습니다. 지금 수행하고 있는 연구도 그 연장선에서, 바이너리 분석 자동화와 프로그램 테스팅을 통해 실제 소프트웨어의 결함을 보다 체계적으로 찾고 검증하는 방법을 연구하고 있습니다.

편: Software Engineering 분야에서 “논문으로서의 성과”와 “실제 시스템에 적용 가능한 기술” 사이의 간극이 종종 지적됩니다. 교수님의 경험을 바탕으로, 이 두 영역을 연결하기 위해 연구자가 가져야 할 관점이나 접근 방식은 무엇이라고 생각하시는지요?

김: 개인적으로 보안 학회와 SE 학회 양쪽에 논문을 투고하면서, 두 커뮤니티가 중요하게 보는 지점에 차이가 있다는 점을 느꼈습니다. 보안 학회에서는 실제 공격 가능성이나 방어 효과처럼 실질적인 임팩트를 상대적으로 중요하게 보는 반면, SE 학회에서는 문제를 바라보는 새로운 관점이나 아이디어의 참신성도 함께 중요하게 평가합니다. 이 차이는 논문성과와 실용성 사이의 간극이 어디서

비롯되는지를 어느 정도 보여준다고 생각합니다. 각 커뮤니티가 강조하는 평가 기준이 다르다 보니, 실용적인 의미가 크더라도 학문적 참신성을 충분히 설명하지 못하면 약하게 보일 수 있고, 반대로 아이디어가 새롭더라도 실제 시스템에서의 효과를 충분히 보여주지 못하면 한계로 지적될 수 있습니다.

저는 참신성과 실용성은 서로 분리된 가치라기보다 함께 고려되어야 할 요소라고 생각합니다. 이 간극을 줄이기 위해서는 실제 시스템에서 발생할 수 있는 예외 상황과 코너 케이스를 포함할 수 있도록 다양한 실험과 심도 깊은 분석이 필요합니다. 특히 실패 사례를 잘 분석하는 과정은 더 좋은 연구 문제를 발굴하는 경로가 되고, 논문으로서도 더 큰 기여를 할 수 있습니다. 이러한 맥락에서 저는 실험 설계 시 데이터셋의 다양성을 중요하게 생각하는 편입니다. 특정 벤치마크나 제한된 환경에서만 잘 동작하는 기술은 실제 시스템으로 확장되기 어렵기 때문에, 가능한 한 다양한 프로그램, 실행 환경, 데이터셋을 포함해 평가하려고 합니다.

결국 논문 성과와 실용성은 서로 대립되는 목표가 아니라고 생각합니다. 현실의 문제를 깊이 이해하고, 그 문제를 학문적으로 일반화할 수 있을 때 좋은 논문도 나오고 실제로 쓸 수 있는 기술도 나온다고 생각합니다.

편: SE 및 시스템 보안 분야를 연구하려는 학생들에게, 지금 시점에서 반드시 갖춰야 할 역량과 연구 태도는 무엇이라고 생각하시는지요? 또한 연구 주제를 선택할 때 어떤 기준을 가장 중요하게 보시는지도 궁금합니다.

김: 다른 분야와 마찬가지로 기초를 깊이 이해하는 능력이 중요하다고 생각합니다. 최근에는 AI 도구나 자동화된 분석 도구가 많이 발전했지만, 시스템 보안이나 low-level SE 연구에서는 운영체제, 컴퓨터 구조, 컴파일러, 프로그램 분석, 메모리 구조 같은 기본 지식이 중요합니다. 특히 보안 연구에서는 코드가 실제로 어떻게 실행되는지 이해하는 것이 핵심입니다. 어셈블리 수준에서 프로그램의 동작을 추적하고, 디버거로 실행 흐름을 직접 확인하며, 도구가 내놓은 결과를 그대로 받아들이지 않고 검증하는 능력이 이 분야에서는 기본기라고 할 수 있습니다.

연구 태도 측면에서는 끝까지 파고드는 태도가 중요합니다. 특히 바이너리 코드 분석을 수행할 때는 예상과 다른 결과가 자주 나옵니다. 그럴 때 단순히 실패로 받아들이는 것이 아니라, 왜 안 되는지, 어떤 가정이 틀렸는지, 어떤 코너 케이스가 있었는지를 끝까지 파악해야 합니다. 이런 과정에서 오히려 좋은 연구 문제가 발견되는 경우가 많습니다.

연구 주제를 선택할 때는 세 가지를 중요하게 봅니다. 첫째, 실제로 중요한 문제인가. 둘째, 기존 방법이 왜 충분하지 않은가. 셋째, 내가 그 문제를 깊이 파고들 수 있는 기술적 기반과 흥미가 있는가입니다. 단순히 유행하는 주제라서 선택하기보다는, 실제 시스템에서 반복적으로 발생하는 문제인지, 그리고 해결되었을 때 개발자나 보안 분석자에게 의미 있는 영향을 줄 수 있는지를 봐야 한다고 생각합니다.

마지막으로 학생들에게 강조하고 싶은 것은, 최신 기술을 빠르게 따라가는

것도 중요하지만 좋은 문제를 스스로 발견하는 능력이 그보다 더 중요하다고 생각합니다. 좋은 문제는 실제 시스템을 만져보고, 도구를 써보고, 실패를 경험하고, 그 한계를 직접 확인하는 과정에서 나오는 경우가 많습니다. 많이 도전해 보시고, 실패하더라도 좌절하지 않으셨으면 합니다. 시스템 보안과 SE 연구에서는 그런 경험 하나하나가 결국 큰 자산이 됩니다.



버그 수를 믿어도 될까? 퍼저 벤치마킹 결과의 무작위성 측정에 관하여



• 이성민
UCLA
seongminlee@sigsoft.org

퍼저 (Fuzzer)는 소프트웨어의 결함을 찾기 위해 무작위 입력을 생성하여 프로그램에 전달하는 테스트 기법이다. 퍼저를 사용하면 개발자가 예상하지 못한 입력에 대해 프로그램이 어떻게 반응하는지 확인할 수 있으며, 이를 통해 잠재적인 버그를 발견할 수 있다. 따라서 퍼저의 성능은 발견된 버그 수를 기준으로 판단하는 것이 가장 직관적이다. 그러나 많은 경우, 퍼저의 성능을 발견된 버그 수 대신 퍼저가 실행해 본 코드의 양, 즉 코드 커버리지 (code coverage)로 측정한다. 버그는 좀처럼 나오지 않아 그 수를 제대로 세려면 막대한 계산 자원이 들기 때문이다. 게다가 실행되지 않은 코드에서는 어차피 버그를 찾을 수 없으니, 커버리지는 버그의 대리 지표 (proxy)가 될 수 있다. 그러다 보니 그동안의 논의는 커버리지가 버그를 얼마나 잘 대신하는지에 집중되어 왔고, 버그 수는 그 대리 지표를 채점하는 정답지처럼 여겨져 왔다. 그러나 퍼저의 성능을 주어진 벤치마크(benchmark), 즉 퍼저를 시험할 대상으로 미리 골라둔 프로그램들에서 단순히 발견된 버그 수로 평가하는 것은 과연 타당한가? 벤치마크에 어떤 프로그램이 들어가는냐에 따라 그 수가 크게 달라질 수 있기 때문이다. 본 글에서는 퍼저 벤치마킹 결과의 무작위성을 측정하는 방법과 그 의미에 대해 논의하고자 한다.

1. 커버리지와 버그, 엇갈리는 순위

퍼저의 성능을 평가하는 절차는 크게 두 가지다. 퍼저가 찾아낸 버그의 수로 순위를 매기는 버그 기반 벤치마킹 (bug-based benchmarking), 그리고 퍼저가 달성한 커버리지로 순위를 매기는 커버리지 기반 벤치마킹 (coverage-based benchmarking)이다. 앞서 언급했듯 실제 연구에서 표준처럼 쓰이는 쪽은 후자다 (Klees et al., 2018; Gligoric et al., 2015). 비용 문제가 있지만, 버그를 세는 일 자체가 간단하지 않기 때문이다. 퍼징이 쏟아 내는 수천 건의 오류 보고서 가운데 어디까지를 같은 버그로 묶을지는 중복 제거 (deduplication) 방식에 좌우되고, 그래서 같은 실험을 하고도 집계 기준에 따라 결론이 달라질 수 있다 (Gavrilov et al., 2020; Schloegel et al., 2024). 반면 커버리지는 수만 개의 분기 단위로 자동 측정되는 촘촘한 신호라, 한 번 돌려서 많아도 몇 건 나오는 버그보다 퍼저 사이의 미세한 차이를 훨씬 잘 가려낸다.

그렇다면 커버리지는 정말로 버그를 대신할 수 있을까? Böhme et al. (2022)는 구글이 운영하는 퍼저 벤치마크 집합 FuzzBench (Metzman et al., 2021)에서 두 절차를 나란히 돌려 이 물음을 확인해 보았다. 절반은 예상대로였다. 커버리지를 많이 달성한 퍼저가 버그도 많이 찾아서, 두 지표의 상관관계 (correlation)는 매우 강했다. 문제는 순위였다. 공부를 오래 한 학생이 시험을 잘 보는 경향은 뚜렷해도 공부 시간 순위와 성적 순위가 그대로 겹치지는 않듯, 상관이 강하다고 해서

순위까지 같아지는 것은 아니다 (Schober et al., 2018). 커버리지로 매긴 순위표와 버그 수로 매긴 순위표를 나란히 놓으면 상당수 퍼저의 자리가 어긋나 있었다. 심한 경우 커버리지에서 1위를 차지한 퍼저가 버그 찾기에서는 최하위로 내려가기도 한다. 이는 사소한 어긋남이 아니다. 같은 퍼저를 같은 데이터로 평가하고도 어느 지표를 택하느냐에 따라 승자가 뒤바뀐다면, 지금까지 “우리 퍼저가 더 낫다”고 주장해 온 수많은 평가 결과는 대체 무엇을 근거로 삼은 것일까.

2. 무작위성은 어떻게 재는가

두 순위는 왜 어긋날까? 본 글의 필자가 공저한 논문 “In Bugs We Trust? On Measuring the Randomness of a Fuzzer Benchmarking Outcome”은 이 물음에서 출발했다 (Madadi et al., 2026). 한 가지 가능한 설명은 버그가 커버리지보다 훨씬 성기게 분포한다는 것이다. 수만 개에 이르는 분기와 달리 버그는 프로그램 전체에 드문드문 흩어져 있어, 어떤 프로그램을 벤치마크로 골랐느냐에 따라 버그 기반 평가의 결과가 크게 달라지기 쉽다. 한쪽 결과가 이렇게 일정하지 않다면, 두 순위가 어긋나는 것도 그 탓일 수 있다. 그렇다면 버그 기반 벤치마킹은 정말 커버리지 기반보다 더 무작위적일까?

우리는 이 무작위성을 직접 재 보기로 했다. 재는 방법으로는 심리학에서 널리 쓰이는 척도인 반분 신뢰도 (split-half reliability)를 빌렸다 (Cortina, 1993). 시험이 믿을 만한지 따질 때 쓰는 값으로, 문항을 두 묶음으로 나눠 따로 채점한 다음 양쪽에서 학생들의 등수가 같게 나오는지를 본다. 등수가 비슷하다면 그 문항들은 재려는 것을 일관되게 재고 있는 셈이고, 처음 보는 문항으로 다시 치러도 등수는 크게 달라지지 않는다. 반대로 양쪽 등수가 크게 어긋난다면, 등수를 가른 것은 문항 구성이다.

퍼저 벤치마킹도 구조가 같다. 퍼저의 성능이라는 직접 볼 수 없는 것을 벤치마크라는 문항으로 재어 순위를 매기기 때문이다. 벤치마크 집합을 무작위로 절반씩 쪼개고, 양쪽에서 각각 퍼저 순위를 매겨 두 순위표를 건주면 된다. 벤치마크 집합을 B , 커버리지를 c , 버그 수를 b 라 하자. 지표 $m \in \{c, b\}$ 로 매긴 순위표를 $R(B, m)$, 두 순위표의 유사도를 ρ 라 쓰면, 지표 m 의 일치도 (concordance)는 다음과 같다.

$$\gamma_m = \mathbb{E}[\rho(R(B_1, m), R(B_2, m))]$$

여기서 B_1 과 B_2 는 B 를 겹치지 않게 반으로 가른 두 부분이고, $\mathbb{E}$ 는 가능한 모든 분할에 대한 평균이다. 유사도 ρ 는 두 순위표가 얼마나 닮았는지를 -1에서 1 사이의 값으로 나타내며, 일치도 γ_m 은 그 평균이다. 0.4에 못 미치면 일치도가 약하다고, 0.4에서 0.69 사이면 보통이라고, 0.7을 넘으면 강하다고 읽는다

(Schober et al., 2018). 커버리지가 버그를 얼마나 대신할 수 있는지도 같은 틀로 잴 수 있다. 한쪽 절반은 커버리지로, 다른 쪽 절반은 버그 수로 순위를 매겨 두 순위표를 견주는 것이다. 이 값을 커버리지 기반과 버그 기반 사이의 동의도 (agreement) ω 라 부른다.

$$\omega = \mathbb{E}[\rho(R(B_1, c), R(B_2, b))]$$

일치도가 양쪽에 같은 지표를 쓰는 데 반해, 동의도는 서로 다른 지표를 쓴다. 우리가 답하려 한 물음은 두 가지다.

RQ1. 버그 기반의 일치도 γ_b 는 커버리지 기반의 γ_c 에 비해 얼마나 되는가? 일치도가 낮은 쪽은 어떤 벤치마크를 고르느냐에 따라 순위가 더 크게 달라진다. γ_b 가 γ_c 보다 작다면, 버그 기반 평가가 커버리지 기반보다 더 무작위적이라는 뜻이다.

RQ2. γ_b 와 ω 가운데 어느 쪽이 더 크다? 버그 기반 평가가 무작위적이라면, 다른 벤치마크에서 나올 버그 순위를 맞히는 데 버그 수보다 커버리지가 나올 수도 있다. 두 값 모두 다른 절반의 버그 순위를 맞히는 문제이고 맞히는 쪽 지표만 다르므로, ω 가 더 크다면 그 짐작이 맞는 셈이다.

우리는 널리 쓰이는 두 퍼징 벤치마크 집합 FuzzBench와 Magma (Hazimeh et al., 2020)를 대상으로 삼았다. 둘은 퍼징 결과에서 버그를 알아내는 방식이 다르다. FuzzBench에는 정해진 버그 목록이 없다. 퍼징이 다 끝난 뒤 쌓인 오류 보고서를 자동으로 묶고 사람이 직접 중복을 걸러 내야, 비로소 서로 다른 버그가 몇 개인지 정해진다 (post-hoc analysis). Magma는 과거에 발견된 버그 138개를 최신 버전 프로그램에 다시 심어 두었다 (forward-porting). 버그마다 그 버그를 일으키는 조건이 충족되면 곧바로 신호를 내는 검사문을 함께 넣어 두어, 퍼저가 어느 버그를 찾았는지 그 자리에서 알 수 있다.

3. 실험 결과

3.1 커버리지 기반과 버그 기반의 일치도

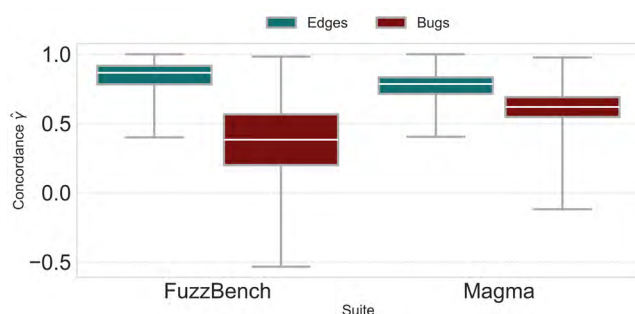


Figure 1. 두 순위표 유사도 ρ 의 분포 (FuzzBench와 Magma, 커버리지와 버그 수).

fig. 1은 벤치마크 집합을 반씩 갈라 본 수많은 경우에 대해 유사도 ρ 가 어떻게 분포하는지를, 지표별로 또 FuzzBench와 Magma 별로 보여 준다. 커버리지 쪽은 값이 높은 자리에 좁게 모여 있고, FuzzBench의 버그 쪽은 아래로 넓게 퍼져 있다. 커버리지 기반의 일치도 γ_c 는 FuzzBench에서 0.84, Magma에서

0.78로 양쪽 모두 강하다. 벤치마크를 어떻게 반으로 가르든 커버리지가 매긴 순위는 거의 그대로였다. 반면 버그 기반의 γ_b 는 FuzzBench에서 0.37, 약한 일치에 머물렀다. 어느 벤치마크가 어느 쪽에 들어가느냐에 따라 버그 순위가 뒤바뀌었다. 버그 기반 평가가 커버리지 기반보다 더 무작위적이라는 뜻이다. 커버리지와 버그의 순위가 어긋난다는 Böhme et al. (2022)의 결과 역시 이 무작위성으로 설명할 수 있다.

흥미로운 쪽은 Magma다. 같은 버그 기반인데도 일치도가 0.60으로 FuzzBench의 0.37과 크게 벌어진다. 버그 기반 평가라도 Magma에서는 훨씬 덜 무작위적이라는 뜻인데, 무엇이 이런 차이를 만들었을까? Magma에는 버그 백에 개가 일부러 심겨 있는 반면, FuzzBench의 버그는 원래 프로그램에 있던 것이라 훨씬 성기게 흩어져 있다. 버그가 촘촘히 깔려 있으면 어느 벤치마크를 골라도 비슷한 순위가 나오므로, 우리는 이 차이가 일치도를 갈랐다고 본다.

3.2 일치도와 동의도의 비교

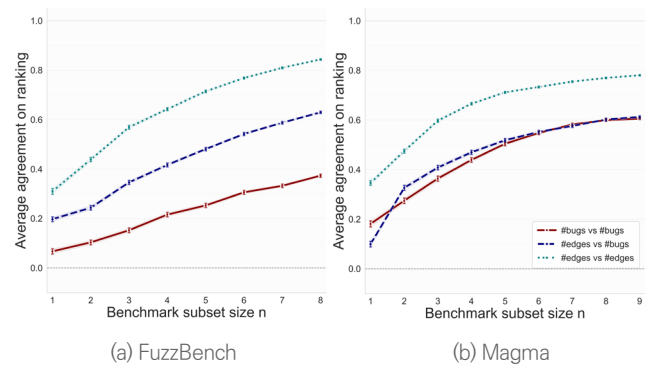


Figure 2. 벤치마크 수에 따른 일치도와 동의도 (초록: γ_c , 빨강: γ_b , 파랑: ω).

버그 기반 평가가 이렇게 무작위적이라면, 다른 벤치마크에서 나올 버그 순위를 예측하는 데 버그 수 말고 커버리지를 쓰면 어떨까. fig. 2는 반분 신뢰도를 잴 때 나누는 각 묶음의 벤치마크 수가 늘어감에 따라 일치도와 동의도가 어떻게 달라지는지를 보여 준다. 초록 선은 커버리지 기반의 일치도 γ_c , 빨강 선은 버그 기반의 γ_b , 파란 선은 동의도 ω 다.

FuzzBench에서는 파란 선이 빨강 선보다 위에 있다. 앞 소절에서 본 γ_b 가 0.37이었던 데 반해 ω 는 0.63으로 거의 두 배다. 커버리지로 매긴 순위가 버그 수로 매긴 순위보다 다른 벤치마크의 버그 순위를 더 잘 맞힌 것이다. 반면 Magma에서는 두 선이 겹쳐 있다. γ_b 가 0.60, ω 가 0.61로 차이가 없다. 두 결과를 가른 것은 버그 기반 평가가 얼마나 무작위적이나다. 버그는 드물게 걸려 결과가 일정하지 않은 반면 커버리지는 퍼저가 실행한 만큼 꾸준히 쌓이므로, 버그 쪽이 무작위적인 곳일수록 커버리지가 더 나은 대리 지표가 된다.

정리. 커버리지 기반의 일치도는 두 집합 모두에서 강했지만, 버그 기반은 둘 다 그보다 약했다. 그 정도를 가른 것은 벤치마크에 버그가 얼마나 촘촘히 놓여 있느냐다. 버그가 성기게 놓일수록 어느 벤치마크를 골랐는지가 결과를 좌우하고, 그칠수록 커버리지가 더 나은 대리 지표가 된다.

4. 그린 퍼징: 벤치마킹의 탄소 발자국

마지막으로, 일치도가 높다는 것은 벤치마크의 일부만 써도 전체와 비슷한 순위가 나온다는 뜻이기도 하다. 그렇다면 일치도가 높을수록 퍼저 벤치마킹을 더 적은 비용으로 돌릴 수 있지 않을까? 이 물음이 중요한 까닭은 퍼저 벤치마킹에 드는 비용이 크기 때문이다. FuzzBench와 Magma 실험은 각각 나무 여섯 그루가 1년 동안 흡수할 이산화탄소를 내놓았고, 이를 줄이려는 그린 퍼징 (green fuzzing) 연구도 이어지고 있다 (Ounjai et al., 2023; Andarzian et al., 2023).

이를 알아보기 위해 우리는 벤치마크의 일부만으로 매긴 퍼저 순위가 전체로 매긴 것과 비슷해지려면 그 일부가 얼마나 커야 하는지를 확인하고, 이를 일치도와 견주었다. 실험 결과, 커버리지 기반은 절반만 남겨도 전체와 매우 강하게 일치했지만 버그 기반은 4분의 3을 남겨야 같은 수준에 이르렀다. 즉 일치도가 높은 쪽이 더 적게 돌리고도 전체를 다 썼을 때와 같은 순위를 얻었고, 탄소 배출은 그만큼 두 배 넘게 줄었다.

5. 맺음말

도구를 공정하게 견주기 위해 연구 분야마다 벤치마크가 만들어진다. 벤치마크가 있어야 비교가 가능하지만, 하나가 자리를 잡고 나면 그 벤치마크에 맞춰 도구를 다듬게 되기 쉽다. 그렇게 얻은 결론은 그 벤치마크 밖으로 넓혀지지 않고 (Zeller and Just, 2019), 지표를 하나만 놓고 도구의 순위를 매기면 그 도구가 지닌 다른 면을 놓친다 (Li et al., 2021). 벤치마크가 내놓은 결과를 읽을 때 주의가 필요한 까닭이다.

본 글은 널리 쓰이는 퍼징 벤치마크에서 버그 기반 평가의 한계를 들여다보았다. 커버리지 기반 결과와 버그 기반 결과가 어긋난다는 데서 출발해, 그 어긋남이 왜 생기는지를 따졌다. 심리학에서 쓰는 반분 신뢰도를 빌려 두 지표의 무작위성을 잴다. 실험 결과, 버그 기반은 무작위성이 컸고 커버리지 기반은 훨씬 일관되었으며, 두 결과가 어긋나는 까닭도 여기에 있었다. 그 무작위성은 벤치마크에 버그가 충분히 갖춰지지 않을수록 커졌고, 그런 경우 커버리지 기반 평가가 버그 기반 평가보다 더 나은 대리 지표였다.

References

Seyed Behnam Andarzian, Cristian Daniele, and Erik Poll. Green-fuzz: Efficient fuzzing for network protocol implementations. In *International Symposium on Foundations and Practice of Security*, pages 253–268. Springer, 2023.

Marcel Böhme, László Szekeres, and Jonathan Metzman. On the reliability of coverage-based fuzzer benchmarking. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 1621–1633, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392211. doi: 10.1145/3510003.3510230. URL <https://doi.org/10.1145/3510003.3510230>.

Jose M Cortina. What is coefficient alpha? an examination of theory and applications. *Journal of applied psychology*, 78 (1):98, 1993.

Miroslav Gavrilov, Kyle Dewey, Alex Groce, Davina Zamanzadeh, and Ben Hardekopf. A practical, principled measure of fuzzer appeal: A preliminary study. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 510–517. IEEE, 2020.

Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, and Darko Marinov. Guidelines for coverage-based comparisons of non-adequate test suites. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(4):1–33, 2015.

Ahmad Hazimeh, Adrian Herrera, and Mathias Payer. Magma: A ground-truth fuzzing benchmark. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), December 2020. doi: 10.1145/3428334. URL <https://doi.org/10.1145/3428334>.

George Klees, Andrew Ruef, Benji Cooper, Shiyi Wei, and Michael Hicks. Evaluating fuzz testing. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 2123–2138, 2018.

Yuwei Li, Shouling Ji, Yuan Chen, Sizhuang Liang, Wei-Han Lee, Yueyao Chen, Chenyang Lyu, Chunming Wu, Raheem Beyah, Peng Cheng, et al. {UNIFUZZ}: A holistic and pragmatic {Metrics-Driven} platform for evaluating fuzzers. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2777–2794, 2021.

Ardi Madadi, Seongmin Lee, Cornelius Aschermann, and Marcel Böhme. In bugs we trust? on measuring the randomness of a fuzzer benchmarking outcome. *Proc. ACM Softw. Eng.*, 3(FSE), June 2026. doi: 10.1145/3797112. URL <https://doi.org/10.1145/3797112>.

Jonathan Metzman, László Szekeres, Laurent Simon, Read Sprabery, and Abhishek Arya. Fuzzbench: an open fuzzer benchmarking platform and service. In *Proceedings of the 29th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, pages 1393–1403, 2021.

Jiradet Ounjai, Valentin Wüstholtz, and Maria Christakis. Green fuzzer benchmarking. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1396–1406, 2023.

Moritz Schloegel, Nils Bars, Nico Schiller, Lukas Bernhard, Tobias Scharnowski, Addison Crump, Arash Ale-Ebrahim, Nicolai Bissantz, Marius Muench, and Thorsten Holz. Sok: Prudent evaluation practices for fuzzing. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1974–1993, San Francisco, CA, USA, 2024. IEEE. doi: 10.1109/SP54263.2024.00137.

Patrick Schober, Christa Boer, and Lothar Schwarte. Correlation coefficients: Appropriate use and interpretation. *Anesthesia & Analgesia*, 126:1, 02 2018. doi: 10.1213/ANE.0000000000002864.

Andreas Zeller and Sascha Just. When results are all that matters: Consequences. <https://fuzzing-survey.org/blog/when-results-are-all-that-matters-2019>. Accessed: 2026-01-11.



ICST 2026 개최 후기



• 김문주



• 홍신

지난 5월 18일부터 22일까지 대전컨벤션센터(DCC)에서 열린 IEEE International Conference on Software Testing, Verification and Validation (ICST) 국제학술대회의 조직위원을 맡아 행사를 개최하였습니다. 학회 유지 제안에서 개최에 이르기까지 2년의 시간은 소프트웨어 테스팅 커뮤니티와 함께 ICST가 학술대회로서 어떤 역할을 해야 하는지 고민한 뜻깊은 시간이었습니다. 중요한 시기에 ICST 커뮤니티를 우리나라에 맞이할 수 있었던 것에 보람을 느낍니다. 아울러, 성공적인 개최를 위해 애써 주신 소프트웨어공학 소사이어티 여러분께 깊이 감사드립니다.

IEEE ICST는 소프트웨어 검증과 테스팅, 신뢰성 연구를 논의하는 학술대회로, 2008년에 시작되어 올해로 19회를 맞았습니다. 산업계와 학계의 실질적인 교류를 중시하는 학회로, 다양한 분야의 산업계 전문가들이 학술대회의 주요 구성원으로 참여한다는 특징이 있습니다. 개방적인 운영을 바탕으로 서로 다른 분야와 배경을 지닌 신진 연구자들이 활발히 참여하며 학회를 이끌어 간다는 점도 ICST의 중요한 특징입니다. 학회가 처음 열린 2008년 노르웨이 대회 때부터 우리나라 여러 연구자들은 Research 트랙에서 꾸준히 논문을 발표해 오셨으며, 현재 ICST 커뮤니티의 주축으로 활동하고 계십니다.

ICST 2026의 Research 트랙에는 34개국에서 총 166편의 논문이 접수되었으며, 이 가운데 44편이 발표 논문으로 선정되었습니다. 166편이라는 접수 규모는, ICST 2011과 함께 역대 최대 기록입니다. Research 트랙 외에도 산업체 트랙과 단편 논문 트랙을 비롯한 10개 트랙에서 71편의 논문이 발표되었습니다. 또한 5월 18일과 22일에 열린 9개 워크숍에서도 총 45편의 논문 발표를 비롯한 다양한 학술 행사가 진행되었습니다.

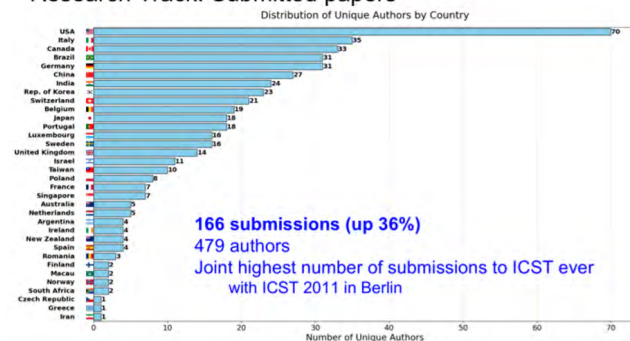
코로나19 사태로 인해 오프라인으로 모일 수 없었으며 소프트웨어 분야 학술대회의 대형화와 논문 모집 방식의 변화가 두드러졌던 2020~2022년을 지나면서, ICST 커뮤니티는 학회의 방향을 두고 많은 고민을 이어 왔습니다. 이탈리아 나폴리에서 개최된 ICST 2025에서는 트랙을 확대해 다양한 방식으로

학술대회에 참여할 수 있도록 하고 저변을 넓히고자 하였으며, 그 결과 참가자 수와 발표 논문 수가 다시 증가하는 성과를 거두었습니다. ICST 2026에서도 이러한 흐름을 이어 가고자 노력하였습니다. 기존의 모든 트랙을 유지하는 한편, 올해에는 Artifact Evaluation 트랙을 새롭게 도입하였습니다. Artifact Evaluation 트랙의 학술위원장을 맡으신 강성민 박사님의 기획으로 순조롭게 출범하였으며, 첫 시도였음에도 Research 트랙의 44편 가운데 25편이 참여할 만큼 큰 호응을 얻었습니다. 이를 통해 재현 가능한 소프트웨어 테스팅 연구에 대한 커뮤니티의 높은 관심을 확인할 수 있었습니다.

이번 프로그램에서 특히 눈여겨볼 점은 우리나라의 젊은 연구자들이 주요 학술 트랙의 위원장으로 활약하셨다는 사실입니다. 산업체 트랙의 안가빈 교수님, Testing Tool & Data Showcase 트랙의 김진한 박사님, 포스터 트랙의 김미정 교수님, Artifact Evaluation 트랙의 강성민 박사님께서 공동학술위원장으로 논문 모집부터 심사와 선정에 이르는 전 과정을 성공적으로 이끌어 주셨습니다. 이와 함께 단편 논문 트랙의 차상길 교수님, Education 트랙의 남재창 교수님, Journal-first 트랙의 배경민 교수님, Doctoral Symposium의 이주용 교수님, 튜토리얼의 김윤호 교수님께서도 각 트랙의 위원장으로 활동하시며 알찬 프로그램을 만들어 주셨습니다.

학술대회를 앞두고 예기치 못한 국제 정세의 변화로 참가에 어려움을 겪는 분들이 있다는 소식을 접해 걱정하기도 하였습니다. 그럼에도 불구하고, 36개국에서 약 300명의 많은 연구자가 참석해 주셨습니다. 전체 일정은 Research 트랙 공동학술위원장인 Neil Walkinshaw 교수님과 Xiaoyuan Xie 교수님께서 편성해 주셨으며, 두 분의 제안에 따라 Research 트랙 논문을 비롯한 일반 논문에는 20분의 발표 시간과 5분의 질의응답 시간을 배정하였습니다. 2020년 이후 온라인 발표와 짧은 발표가 늘고, 대형 학술대회에서 포스터 발표가 중심이 된 상황에서, 이러한 25분 발표 구성은 오히려 새롭게 다가왔습니다. 발표자들은 충실히 준비하였고, 참가자들도 경청하며 활발하게 토론에 참여해 주셨습니다.

Research Track: Submitted papers



Research Track: Accepted papers

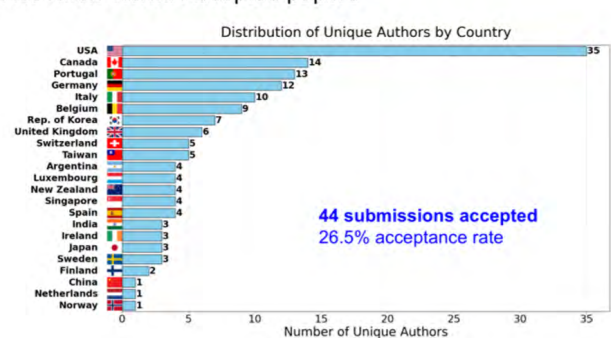


그림 1. Research 트랙에 투고된 논문과 발표 선정된 논문의 통계



그림 2. 학회 개최 선언을 하는 공동 조직위원장 김문주 교수



그림 3. 리셉션 참가자

메인 컨퍼런스 기간 매일 오전에 진행된 기조연설은 이번 학술대회의 하이라이트였습니다.

첫 번째 키노트에서는 MPI-SP의 Marcel Böhme 교수님께서 “Adversarial Benchmarking: An Urgent Call for a Stronger Signal of Technological Progress”라는 제목으로 발표하셨습니다. Böhme 교수님께서 AI를 비롯한 여러 연구 분야에서 부적절한 벤치마킹이 초래하는 위험을 환기하고, 소프트웨어를 대상으로 한 임상시험과 같은 엄밀한 평가를 설계하는 데 전문성을 지닌 소프트웨어 테스팅 연구자들의 역할을 강조하셨습니다. 두 번째 키노트는 Adaptive Random Testing과 Metamorphic Testing의 개념을 창안하신 연구자로 유명하신 T. Y. Chen 교수님께서, 현재 집중하고 계신 Fault-based Testing의 개념과 주요 연구 문제를 소개하셨습니다. Chen 교수님께서 학술행사의 전 일정에 참여하셨으며, 틈틈이 여러 연구자 및 학생들과 공동 연구를 제안하고 논의를 이어 가는 열정적인 모습을 보여 주셨습니다. 탁월한 연구 업적을 이룬 석학의 이러한 자세는 커뮤니티 구성원들에게 깊은 인상과 도전을 주었습니다. 세 번째 키노트는 KAIST의 유신 교수님께서 “Keeping up

with the Abstraction”이라는 제목으로, LLM과 AI 에이전트를 위한 새로운 소프트웨어 계층을 탐구하고 구축해 나가자고 제안하는 도전적인 발표를 해주셨습니다. 세 분의 기조연사 모두 학술대회의 전 일정에 함께하셨으며, 많은 참가자가 키노트 발표자들과 토론을 이어 가는 모습을 볼 수 있었습니다.

5월 20일 오후 마지막 시간에 진행된 Open SC Meeting에서는 운영위원들과 참가자들이 함께 모여 ICST가 앞으로 나아갈 방향을 두고 자유롭게 의견을 나누었습니다. 현재 올바른 방향으로 나아가고 있는지, ICST의 강점을 더욱 발전시키기 위해 무엇을 해야 하는지, 산업계의 참여를 확대하려면 어떤 노력이 필요한지 등을 함께 논의하였습니다. 아울러 앞으로 시도할 수 있는 새로운 프로그램과 다른 학술대회와의 일정 및 역할 조정 등 향후 과제에 대해서도 고민하였습니다.

외국인 참가자들에게 익숙한 서울이 아닌 대전에서 학술대회를 개최하는데 부담이 있었던 것도 사실입니다. 그러나 대전컨벤션센터가 위치한 대덕연구개발특구의 국제적이고 학술적인 환경은 국제 학술대회를 개최하기에 매우 적합하였습니다. 좋은 호텔, 편의 시설을 갖추고 있을 뿐 아니라 갑천변의 자연환경까지 갖추고 있어 참가자들로부터 많은 호평을 받았습니다. 앞으로 대전에서 더 많은 국제학술행사가 개최되기를 기대합니다.

끝으로 ICST를 후원해 주시고 성공적인 개최를 위해 애써 주신 모든 분께 다시 한번 감사드립니다. 학회 유치부터 개최에 이르는 전 과정을 세심하게 살펴 주시고 힘써 주신 고인영 교수님, 백종문 교수님, 유신 교수님께 깊이 감사드립니다. 복잡한 학회 준비 과정을 원활하게 관리하고 행사 기간에는 해외 참가자들을 따뜻하게 맞아 주신 제니컴의 김아영 본부장님과 최지수 대리님, 그리고 준비에 참여해 주신 여러 임직원들도 감사드립니다. 아울러 산업체 후원사인 SAP, 삼성전자, LG전자, Google, 슈어소프트테크와 학술대회 개최를 지원해 주신 대전관광공사, 한국관광공사, KAIST에도 진심으로 감사드립니다.



그림 5. 저녁만찬에서 참가자들이 함께 한 강강술래



그림 4. 키노트 발표(왼쪽부터, Marcel Böhme 교수님, T. Y. Chen 교수님, 유신 교수님)



ICSE 2026 참가 후기

• 김영재, 한승헌
울산과학기술원

올해 4월 12일부터 19일까지 International Conference on Software Engineering (ICSE 2026)에 다녀왔습니다. 브라질 리우 데 자네이루에서 열린 이번 학회에는 전 세계의 수많은 소프트웨어 공학자들이 모여 연구 성과를 공유하고 다양한 이야기를 나눌 수 있었습니다.

올해 ICSE에서 가장 눈에 띄는 주제는 역시 AI였습니다. 기초연설부터 각종 연구 발표, 심지어 바이트 코딩에 대한 튜토리얼까지, AI에 대한 언급이 없는 세션을 찾기 어려울 정도였습니다.

흥미로운 점은 대부분의 연구가 1~2년 전 GPT-4 시절의 모델을 대상으로 하고 있었다는 것입니다. 당시에는 부족한 컨텍스트와 잦은 환각으로 인해 코드 작성을 온전히 맡기기엔 부족하다는 인식이 있었지만, 지금은 코드 작성뿐 아니라 수정, 테스트 작성, 디버깅까지도 AI가 수행할 수 있는 수준에 이르렀습니다.

첫날 기초연설의 핵심 메시지도 여기에 있었습니다. 앞으로 소프트웨어 공학의 병목은 코드 작성이 아니라 검증으로 이동하리라는 것이었습니다. AI에게 내부 구현 방식을 일일이 설명하는 것은 비효율적이므로, 기계적으로 검증 가능한 형태의 기준이 주어진다면 AI는 그 기준을 통과할 때까지 프로그램을 스스로 수정해 나갈 수 있습니다. 다만 요구사항이 충족되었음을 검증하기 위한 기준 자체를 마련하는 일에는 결국 사람의 관여가 필요하다는 점도 함께 강조되었습니다.

ICSE에서 주목할 만한 또 하나의 주제는 소프트웨어 공급망이었습니다. 둘째 날 기초연설이 특히 기억에 남는데, NPM과 같은 패키지 매니저를 통한 공격이나 오픈소스 관리자를 겨냥한 공격 사례를 통해, 의존성과 자동화된 배포가

늘어나면서 공격 표면은 커졌지만 이를 검증하는 방법은 여전히 미흡하다는 문제의식을 다뤘습니다. 해시 검증, 버전 고정, 지연 업데이트와 같은 방법들이 활용되고 있지만, 이들 방식은 결국 “누군가 이미 검증했을 것”이라는 신뢰를 전제로 작동합니다. 시가 코드 생산 속도를 한층 더 높인다면, 이러한 신뢰 체계 역시 함께 재검토해야 하지 않을까 하는 생각이 들었습니다.

이번 ICSE는 한국에서 방문하기 쉽지 않은 브라질에서 개최되어, 개최지 주변 도시를 둘러보는 것도 큰 즐거움이었습니다. 학회가 열린 Barra da Tijuca 지역은 넓은 해변을 끼고 있으면서 리우 데 자네이루의 유명 랜드마크들과도 가까워, 인근 지역을 탐방하기에 편리했습니다. 해변 주변으로 우뚝 솟은 이국적인 산들도 이색적인 풍경을 자아냈습니다.

Barra da Tijuca에는 스테이크하우스와 같은 서양식 식당이 많았습니다. 호텔 근처 식당에서 커다란 스테이크를 꼬치에 꽂아 들고 다니며 즉석에서 잘라주던 모습이 특히 기억에 남습니다.

저희는 이전에 ISSTA와 ASE에도 다녀온 적이 있지만, ICSE에서는 훨씬 더 다양한 사람들과 연구를 접할 수 있었습니다. 특히 정말 많은 연구자들이 AI를 활용한 연구에 관심을 갖고 있으며, AI 시대에 앞으로의 연구를 어떻게 이어가야 할지에 대해 깊이 고민하고 있음을 확인할 수 있었습니다.

지금까지 국제 학회에 몇 차례 참석한 경험은 있었지만, ICSE와 같은 대규모 학회는 이번이 처음이었습니다. 이번 기회를 통해 더 좋은 연구를 진행하여, 전 세계 수많은 연구자들 앞에서 저희 연구를 선보일 수 있는 기회를 더욱 갖고 싶다는 바람을 갖게 되었습니다.



〈사진 1: 뱅콧〉



〈사진 2: ICSE에서 발표〉

제14회 소프트웨어공학 단기전문가강좌를 마치고



• 지은경

KAIST/단기전문가강좌 프로그램위원장

한국정보과학회 소프트웨어공학 소사이어티는 지난 8월 10일(월)부터 11일(화)까지 KAIST 도곡캠퍼스에서 제14회 소프트웨어공학 단기전문가강좌를 개최하였다. 올해 강좌는 「AI 증강 명세 주도 개발과 하네스 엔지니어링(AI-Augmented Spec-Driven Development and Harness Engineering)」을 주제로, 생성형 AI와 AI 에이전트의 발전에 따른 소프트웨어 개발 패러다임의 변화를 살펴보고 이를 실제 개발 과정에 적용해 보는 실습 중심의 교육으로 진행되었다.

최근 AI는 코드 생성을 넘어 요구사항 분석, 설계, 구현, 테스트 및 검증 등 소프트웨어 개발 전 과정에서 활용되고 있다. 이에 따라 소프트웨어 엔지니어에게는 직접 코드를 작성하는 능력뿐 아니라 문제를 구조화하고 명세를 설계하며, AI의 실행을 통제하고 결과의 품질을 검증하는 역량이 더욱 중요해지고 있다. 이번 강좌에서는 이러한 변화에 대응하기 위한 AI 문해력, 명세 기반 산출물 관리, 하네스 엔지니어링, 그리고 명세→생성→검증으로 이어지는 AI 증강 개발 워크플로우를 다루었다.

특히 이번 강좌는 산업 현장에서 AI 기반 개발과 엔지니어링 프로세스를 실천하고 있는 (주)솔루션링크의 전문가들(백영민 책임연구원, 현신영 선임연구원, 조미현 선임 컨설턴트, 김한수 선임 컨설턴트) 강사로 모시고, AI 기술 자체뿐 아니라 기존 소프트웨어공학 원칙을 AI와 어떻게 결합하여 실제 개발에 적용할 것인가에 초점을 맞추었다.

이번 강좌에는 학생 27명과 교수, 연구소, 산업체 관계자를 포함한 일반 13명 등 총 40명이 참가하였다. 당초 35명으로 계획하였으나 AI와 소프트웨어공학에 대한 높은 관심으로 사전 등록이 조기 마감되는 등 참가자들의 관심이 매우 뜨거웠다.

첫째 날에는 AI 기술의 현재와 미래를 살펴보고, 직접 AI를 활용해 소프트웨어를 개발하는 경험을 시작하였다. 백영민 책임연구원이 AI 기술 및 시장 동향을 소개하고, 참가자들은 실습 환경을 구축하여 최소한의 스펙만으로 정적 웹 애플리케이션을 개발하였다. 이어 조미현 선임 컨설턴트가 하네스 엔지니어링의

개념과 역할을 설명하고, 현신영 선임연구원의 폴스택 웹 애플리케이션 스캐폴딩 강의와 실습 진행으로 참가자들은 백엔드, 프론트엔드, 데이터베이스를 포함한 폴스택 개발환경을 구성하였다.

둘째 날에는 구축한 환경에서 실제 애플리케이션을 완성해 가며 AI 증강 명세 주도 개발의 전 과정을 실습하였다. 현신영 선임연구원의 진행으로 데이터베이스 연동 환경을 구성하고 기본적인 데이터 조회/등록/수정/삭제 기능을 구현하여 애플리케이션의 기본 구조를 완성한 후, 김한수 선임 컨설턴트가 요구사항과 데이터를 분석하고 애플리케이션을 설계하는 과정을 이끌었다. 이어 조미현 선임 컨설턴트가 AI 에이전트의 역할과 작업 수행 규칙을 설정하여 명세 기반 구현 환경을 구성하고, AI를 활용한 기능 구현을 진행하였으며, 마지막으로 김한수 선임 컨설턴트와 백영민 책임연구원이 추가 요구사항과 스펙 변경을 반영하는 과정을 함께 진행하였다. 이를 통해 참가자들은 명세를 중심으로 AI를 활용해 요구사항 분석부터 설계, 구현, 검증 및 변경까지 수행하는 AI 증강 개발 프로세스를 직접 경험하였으며, AI의 높은 실행력을 효과적으로 활용하기 위해서는 명확한 명세와 체계적인 통제 및 검증이 중요하다는 점을 살펴볼 수 있었다.

강좌 종료 후 실시한 설문에서 참가자들의 높은 만족도를 확인할 수 있었다. 교육 전체 만족도는 5점 만점에 4.8점 이상으로 나타났으며, 강의 내용과 실습 중심의 구성, 그리고 AI 시대의 새로운 소프트웨어공학 방법론을 직접 경험할 수 있었던 점에 대해 참가자들이 긍정적으로 평가한 것으로 나타났다.

이번 제14회 단기전문가강좌는 AI가 소프트웨어 개발 과정에 빠르게 자리 잡고 있는 가운데, 소프트웨어 엔지니어링의 역할과 방법론이 앞으로 어떻게 변화해야 하는지를 함께 고민하는 뜻깊은 자리였다. 소프트웨어공학 소사이어티는 앞으로도 최신 기술 변화와 산업 현장의 요구를 반영한 전문 교육을 지속적으로 마련하여, 연구자와 학생, 산업계 전문가가 함께 배우고 교류할 수 있는 기회를 제공할 예정이다.



제14회 소프트웨어공학 단기전문가강좌 수강 후기



• KHAREL AROGYA (아로겨), 박사과정
KAIST 전산학부

Q1. 간단히 자기소개와 연구 분야를 소개해 주시겠어요?

안녕하세요. KAIST 전산학부 웹 공학 연구실에서 박사과정을 밟고 있는 KHAREL AROGYA(아로겨)입니다. 소프트웨어공학·AI·로보틱스가 만나는 지점에서 연구하고 있습니다. 제 연구는 AI가 생성하거나 수정한 로봇 행동이 단순히 그럴듯해 보이거나 겉보기에는 잘 실행되는 것을 넘어서, 실제로 사용자가 의도한 바와 일치하도록 만드는 데 초점을 맞추고 있어요.

AI-DLC, 즉 「**AI 증강 명세 주도 개발과 하네스 엔지니어링**」 과정에 관심을 갖게 된 것도 이런 이유였습니다. 다루는 대상은 로보틱스가 아니라 소프트웨어 개발이었지만, 그 안에 깔린 엔지니어링 질문들은 놀랍도록 익숙하게 느껴졌거든요.

Q2. AI-DLC 과정을 수강하게 된 계기는 무엇인가요?

이미 연구 프로토타입 작업에 코딩 에이전트를 자주 쓰고 있었기 때문에, 프롬프트 작성법 같은 기초를 배우려던 건 아니었어요. 제가 원했던 건 에이전트를 다루는 좀 더 체계적인 엔지니어링 접근법이었습니다. 에이전트가 개발 과정의 상당 부분을 맡게 되는 순간, “어떤 모델이 더 나은가”, “프롬프트를 어떻게 써야 하는가” 같은 질문만으로는 부족해지거든요.

이 과정이 흥미로웠던 건 **AI 증강 명세 주도 개발**을 단순히 AI에게 코드를 생성시키는 방식이 아니라, 명세·컨텍스트·구현·검증·요구사항 변화, 그리고 에이전트를 둘러싼 환경까지 아우르는 하나의 엔지니어링 프로세스로 다뤘다는 점이었습니다.

Q3. 과정에서 가장 유용했거나 기억에 남는 개념은 무엇인가요?

Harness Engineering입니다.

강의 중 “What is a harness?”라는 질문으로 시작했는데, 얼마 지나지 않아 “A better question, what is not a harness?”라는 질문으로 넘어갔던 게 인상 깊었어요. 관점을 뒤집으니 오히려 명확해지더라고요.

같은 에이전트라도 주변 환경에 따라 놀랍도록 믿음직스러워 보이기도 하고, 완전히 rogue가 되어버리기도 하거든요. 모호한 지시와 부족한 검증 속에서는 행동을 예측하기 어렵지만, 명세·관련 지식·도구·제약·검증 장치가 잘 갖춰진 환경에서는 같은 에이전트가 훨씬 체계적으로 움직입니다. 그 주변을 의도적으로 구성한 시스템이 바로 harness인 거죠.

Skills, Subagent, Hook 같은 구체적인 메커니즘들을 실습했지만, 정작 제 안에 남은 건 더 큰 아이디어였어요. 에이전트 자체는 시스템을 구성하는 하나의 요소일 뿐이라는 것, 그리고 결과물의 품질을 신경 쓴다면 에이전트를 둘러싼 시스템 자체를 설계해야 한다는 것이었습니다.

Q4. 이 과정을 듣고 나서 소프트웨어 엔지니어링을 대하는 방식에서 가장 크게 바뀔 것 같은 부분은 무엇인가요?

앞으로는 에이전트와의 상호작용 자체에만 집중하기보다, 에이전트가 어떤 환경 안에서 일하게 할지를 먼저 설계하려고 합니다.

“이 에이전트에게서 더 나은 답을 어떻게 끌어낼까”라는 질문만 하기보다, 이 에이전트가 어떤 명세를 기반으로 작업하는지, 어떤 지식을 지속적으로 참고할 수 있어야 하는지, 어떤 결정까지 맡길 수 있는지, 항상 지켜야 할 제약은 무엇인지, 변경 전후에 자동으로 실행되어야 할 것은 무엇인지, 그리고 결과가 원래 의도에 맞는지 어떻게 검증할지를 더 많이 고민하게 될 것 같아요.

제게는 이게 가장 큰 사고방식의 변화인 것 같습니다. 더 강력한 에이전트는 물론 유용하지만, 에이전트가 강력해진다고 해서 시스템이 저절로 잘 설계되는 건 아니니까요.

Q5. 이 과정과 박사 연구 사이에서 가장 강하게 연결된다고 느끼는 지점은 어디인가요?

AI 구성요소 하나만 믿기보다, 그 주변에 구조를 설계한다는 아이디어가 가장 강하게 맞닿아 있습니다.

AI 기반 소프트웨어 개발에서는 개발자에게 어떤 의도나 요구사항이 있고, 에이전트가 소프트웨어를 생성하거나 수정합니다. 그리고 그 변경이 실제 요구사항에서 벗어나지 않도록 제어하고 검증하는 구조가 필요합니다.

제 연구도 구조가 비슷해요. 사용자가 의도를 표현하면 AI 모델이 이를 로봇 행동으로 번역하고, 이후 환경 변화에 따라 그 행동이 다시 조정되는 과정을 거치니까요.

두 경우 모두 결과물이 그럴듯해 보이거나 심지어 잘 작동하는 것처럼 보여도, 원래 의도했던 바에서 슬며시 벗어나 있을 수 있습니다.

그래서 이 과정을 들으며 harness라는 개념을 코딩 에이전트 너머로 일반화할 수 있을지 고민하게 됐어요. 로봇의 경우라면, 명시적으로 표현된 요구사항, 무엇을 바꿀 수 있는지에 대한 맥락적 제약, 그리고 어떤 조정이 원래 의도를 위반하는 순간을 감지하는 검증 장치 같은 것들이 그 주변 시스템에 해당할 것 같습니다.

다루는 대상은 다르지만 — 한쪽은 소프트웨어, 다른 한쪽은 로봇 행동이죠. 하지만 원칙은 매우 비슷하다고 느껴요. 에이전트 혼자서 알아서 잘 해내리라 기대하기보다는, 의도한 결과가 시스템 구조상 자연스럽게 나오도록 그 주변을 설계하는 것 말이에요.



기업탐방 특집 - 솔루션링크

AI 시대, 소프트웨어공학의 가치를 다시 묻다

AI Engineering과 Digital Safety Engineering으로 미래 산업을 설계하는 솔루션링크

“AI의 경쟁력은 더 뛰어난 모델이 아니라, 산업과 사회가 신뢰할 수 있는 시스템을 만드는 공학에서 시작된다.”



AI 시대, 다시 소프트웨어공학을 이야기하는 이유

2022년 생성형 AI의 등장은 소프트웨어 산업의 패러다임을 근본적으로 바꾸어 놓았다. ChatGPT를 시작으로 거대언어모델(LLM)이 대중화되었고, 이제 AI는 단순한 생산성 향상 도구를 넘어 기업의 비즈니스 모델과 산업 구조를 재편하는 핵심 기술로 자리 잡고 있다. 제조 현장은 AI Factory를 지향하고 있으며, 자동차 산업은 Software Defined Vehicle(SDV) 시대로 빠르게 전환되고 있다. 휴머노이드 로봇과 자율주행 시스템은 연구 단계를 넘어 실제 산업 현장으로 진입하고 있으며, 금융·의료·국방·에너지와 같은 사회 핵심 인프라에도 AI 도입이 본격화되고 있다.

이처럼 AI 활용이 폭발적으로 증가하는 가운데 산업계의 관심은 자연스럽게 “어떤 AI 모델이 더 뛰어난가”에 집중되는 듯 보인다. 그러나 현장에서 대규모 시스템을 설계하고 운영하는 엔지니어들의 시각은 조금 다르다. 실제 산업에서 중요한 것은 AI 모델 자체보다 AI를 신뢰할 수 있는 시스템으로 구현하는 능력이기 때문이다.

생성형 AI는 기존의 규칙 기반 소프트웨어와 근본적으로 다른 특성을 갖는다. 동일한 입력에 대해 항상 같은 결과를 반환하는 전통적인 소프트웨어와 달리, 생성형 AI는 확률적 특성을 기반으로 다양한 결과를 생성한다. 이러한 특성은 뛰어난 창의성과 유연성을 제공하지만, 동시에 예측 가능성과 검증 가능성이라는 측면에서는 새로운 과제를 제시한다. 산업 현장에서 AI를 활용하기 위해서는 모델의 성능뿐 아니라 요구사항 정의, 위험 분석, 검증, 안전성 확보, 운영 체계, 책임성까지 포함한 새로운 공학적 접근이 필요해진 이유다.

최근 국제사회가 AI Governance를 중요한 화두로 다루는 것도 같은 맥락이다. 유럽연합(EU)의 AI Act, ISO/IEC JTC 1/SC 42의 AI 국제표준, IEEE의 AI 윤리 및 안전 표준, 미국 NIST의 AI Risk Management Framework(AI RMF) 등은 모두 AI 기술 자체를 규제하기 위한 것이 아니라, 산업과 사회가 신뢰할 수 있는 AI를 구현하기 위한 공학적 기반을 마련하는 데 목적이 있다.

이러한 흐름은 하나의 중요한 사실을 보여준다. AI 시대가 도래할수록 오히려 소프트웨어공학(Software Engineering)의 가치가 더욱 커지고 있다는 점이다.

AI는 코드를 생성할 수 있지만 무엇을 개발해야 하는지 정의하지는 못한다. AI는 프로그램을 작성할 수 있지만 시스템 전체의 안전성을 보장하지는 못한다. AI는 기능을 구현할 수 있지만 수백 개의 구성 요소가 복합적으로 동작하는 산업 시스템을 설계하는 것은 여전히 엔지니어의 몫이다. 결국 AI는 소프트웨어공학을 대체하는 기술이 아니라, 소프트웨어공학 위에서 비로소 가치를 발휘하는 기술인 셈이다.

AI 경쟁력은 ‘모델’이 아니라 ‘엔지니어링’이다

최근 기업들은 앞다투어 AX(AI Transformation)를 선언하고 있다. 그러나 AX를 단순히 생성형 AI를 업무에 적용하는 프로젝트로 이해한다면 기대만큼의 성과를 얻기 어렵다.

실제로 많은 기업들은 AI 도입 초기에는 문서 작성, 코드 생성, 검색과 같은 생산성 향상에 만족하지만, 시간이 지날수록 보다 근본적인 질문에 직면하게 된다.



- AI를 기존 업무 프로세스와 어떻게 연결할 것인가?
- AI가 생성한 결과를 어떻게 검증할 것인가?
- AI가 잘못된 판단을 했을 때 누가 책임질 것인가?
- AI를 지속적으로 개선하고 관리할 조직 체계는 어떻게 구축할 것인가?

이 질문들은 모두 기술이 아니라 엔지니어링의 영역이다.

AI 시대의 경쟁력은 더 우수한 모델을 확보하는 것에서 끝나지 않는다. AI를 기업의 업무 프로세스 속에서 안전하고 지속 가능하게 활용할 수 있도록 설계하는 능력이 더욱 중요하다. 이는 요구사항 관리, 프로젝트 관리, 형상관리, 품질관리, 위험관리, 시험·검증, 운영과 유지보수까지 포함하는 소프트웨어공학의 전통적인 영역과 정확히 맞닿아 있다.

최근 학계와 산업계에서 AI Engineering이라는 개념이 주목받는 것도 같은 이유다. AI Engineering은 AI 모델을 개발하는 기술이 아니라, AI를 하나의 산업 시스템으로 설계하고 구축하며 운영하기 위한 공학적 방법론을 의미한다. 즉, AI 모델은 하나의 구성 요소(Component)에 불과하며, 그 위에는 시스템 엔지니어링, 안전공학, 프로젝트 관리, 품질보증, 국제표준, 그리고 조직의 경험과 지식을 축적하는 체계가 함께 존재해야 한다.

이러한 관점은 소프트웨어공학이 오랫동안 강조해 온 생애주기(Life Cycle) 중심 접근과도 정확히 일치한다. 요구사항 분석에서 시작해 설계, 구현, 시험, 배포, 운영, 개선으로 이어지는 전 과정이 유기적으로 연결될 때 비로소 신뢰할 수 있는 AI 시스템이 완성될 수 있다.

25년 전부터 같은 질문을 던져온 기업

이처럼 AI 시대가 되면서 다시 주목받고 있는 소프트웨어공학의 가치를, 25년 전부터 꾸준히 산업 현장에 적용해 온 기업이 있다. 바로 ㈜솔루션링크다.

2000년 설립된 솔루션링크는 국내에서 드물게 소프트웨어공학을 핵심 전문 분야로 출발한 엔지니어링 기업이다. 당시 국내 IT 산업은 개발 인력과 프로젝트 규모 확대에 집중하던 시기였지만, 솔루션링크는 일찍부터 “좋은 소프트웨어는 우수한 개발자만으로 만들어지지 않는다”는

철학 아래 소프트웨어공학과 시스템공학의 산업 적용에 집중했다.

회사는 요구사항공학(Requirements Engineering), 프로젝트 엔지니어링(Project Engineering), 소프트웨어 품질공학, 시스템 엔지니어링, 기능안전, Automotive SPICE, 시험·검증, 소프트웨어 안전공학 등 다양한 분야에서 전문성을 축적하며 금융, 자동차, 공공, 국방, 제조 등 여러 산업에서 엔지니어링 서비스를 제공해 왔다. 이러한 경험은 단순한 프로젝트 수행을 넘어, 산업 현장에서 반복적으로 발생하는 문제를 공학적 방법론으로 해결하고 조직의 역량으로 축적하는 데 초점을 맞추고 있다.

지난 25년간 수행한 프로젝트는 약 1,000여 건에 이르며, 이러한 현장 경험은 이후 솔루션링크의 대표 플랫폼인 ITSCOPE와 AI Engineering 방법론의 기반이 되었다.

오늘날 많은 기업들이 AI 시대의 새로운 개발 방식을 고민하고 있지만, 솔루션링크는 AI 이전부터 축적해 온 소프트웨어공학의 원칙을 바탕으로 AI 시대에 필요한 새로운 엔지니어링 체계를 제시하고 있다. 이들이 말하는 핵심은 단순하다.

“AI는 기술이지만, 신뢰는 공학으로 만들어진다.”



이 철학은 앞으로 살펴볼 솔루션링크에서 제공하는 서비스와 솔루션인 AI Engineering, Digital Safety Engineering, 그리고 ITSCOPE 플랫폼에서 더욱 구체적으로 드러난다.

AI는 사람을 대체하는 것이 아니라 조직을 진화시킨다

많은 기업이 AI를 업무 자동화 도구로 바라본다. 문서를 작성하고, 코드를 생성하고, 회의 내용을 요약하는 기능만으로도 충분한 생산성 향상을 경험할 수 있기 때문이다.

그러나 솔루션링크는 이러한 활용을 AX(AI Transformation)의 시작 단계로 본다.

회사가 생각하는 AX는 단순히 AI를 사용하는 것이 아니라, 조직의 업무 수행 방식 자체를 AI 중심으로 재설계하는 것이다.

이를 위해서는 먼저 현재의 업무 프로세스를 공학적으로 분석해야 한다. 반복되는 업무는 무엇인지, 어떤 의사결정이 중요한지, 어디에서 오류가 발생하는지, 어떤 지식이 반복적으로 활용되는지 이해하지 못하면 AI를 적용하더라도 기대한 효과를 얻기 어렵다.

민상윤 대표는 이를 다음과 같이 설명한다.

“AX의 목표는 AI를 도입하는 것이 아니라 기업의 사업 수행 능력을 혁신하는 것이다. AI는 혁신의 목적이 아니라 혁신을 실현하는 수단이다.”

이러한 관점은 최근 AI 프로젝트에서 자주 언급되는 ‘Human-in-the-Loop’, ‘AI-Augmented Engineering’과도 맥을 같이한다. AI가 인간을 대신하는 것이 아니라, 엔지니어가 더 높은 수준의 판단과 설계에 집중할 수 있도록 지원하는 구조를 만드는 것이 핵심이라는 의미다. 이를 위해서는 도메인별 AX 전환을 위한 체계적인 AI-oriented business process reengineering이 필수이다. 솔루션링크는 법, 제도부터 방법론, 기법에 이르기까지 다각적인 방법론을 연구개발 및 제공하고 있다. 민상윤 대표는 “AX는 결국 비즈니스 프로세스의 재공학의 한 분야인데, 현재 구체적이고 견고한 방법을 제시하는 기업은 아직은 없다. 그만큼 소프트웨어 공학, 조직 관리, 지능화, 데이터 확보 및 변환 등 매우 복잡한 분야이다” 라고 언급했다.

Digital Safety Engineering이라는 새로운 패러다임

최근 산업계에서는 ‘AI Safety’라는 용어가 자주 사용된다. 하지만 솔루션링크는 이보다 더 넓은 개념으로 Digital Safety Engineering을 제시하고 있다.

오늘날 산업 시스템은 단순한 소프트웨어가 아니다. 하나의 시스템 안에는 AI, 소프트웨어, 센서, 클라우드, 네트워크, 로봇, 사이버보안, 기능안전이 동시에 존재한다.

이러한 복합 시스템에서는 어느 하나의 기술만으로는 안전성을 확보할 수 없다.

예를 들어 휴머노이드 로봇을 생각해 보자. 물체를 인식하는 AI가 99%의 정확도를 가진다고 해서 로봇 전체가 안전한 것은 아니다. 이동 제어, 충돌 회피, 힘 제어, 네트워크 지연, 전원 이상, 센서 오차 등 수많은 요소가 함께 고려되어야 한다.

즉, AI Safety만으로는 충분하지 않다.

소프트웨어 안전공학, 시스템 안전공학, 기능안전, 사이버보안, 프로젝트 엔지니어링을 통합적으로 고려하는 새로운 접근이 필요하며, 솔루션링크는 이를 Digital Safety Engineering이라고 정의한다.

이 개념은 단순한 기술이 아니라 디지털 사회를 구성하는 시스템 전체를 안전하게 설계하고 운영하기 위한 공학적 방법론이다.

솔루션링크는 기존의 전장부품의 안전공학과 시스템 안전공학 노하우를 기반으로 AI가 융합된 디지털 시스템에 대한 매우 포괄적인 안전공학 서비스를 제공하고 있다.

프로젝트를 관리하는 것이 아니라, 조직의 지식을 설계하다

프로젝트 관리의 패러다임을 바꾸다...

프로젝트는 끝나지만, 경험은 남아야 한다.

너무도 당연한 이야기처럼 들리지만, 대부분의 조직은 여전히 이 단순한 원칙을 실현하지 못하고 있다. 프로젝트가 종료되면 산출물은 문서함이나 파일 서버 어딘가에 보관되고, 프로젝트를 수행했던 사람들은 다른 업무로 이동한다. 몇 년 후 유사한 프로젝트가 시작되면 새로운 팀은 처음부터 다시 고민하고, 같은 실수를 반복하며, 이미 해결했던 문제를 다시 해결하는 데 시간을 소비한다.

소프트웨어 산업은 제조업과 달리 ‘지식을 생산하는 산업’이다. 자동차를 생산하는 공장은 동일한 공정을 반복하지만, 소프트웨어 프로젝트는 매번 새로운 문제를 해결해야 한다. 그렇기 때문에 프로젝트의 가장 중요한 산출물은 프로그램 코드만이 아니라, 프로젝트 수행 과정에서 축적된 엔지니어링 지식(Engineering Knowledge)이다.

솔루션링크는 오래전부터 이러한 문제를 고민해 왔다. 프로젝트 관리 시스템(PMS)을 단순히 일정과 업무를 관리하는 도구가 아니라, 조직의 경험을 구조화하고 재사용하는 지식 플랫폼(Knowledge Platform)으로 발전시켜야 한다는 것이 회사의 철학이다.

이러한 고민의 결과물이 바로 ITSCOPE이다.

ITSCOPE는 왜 일반 PMS와 다른가

국내에는 다양한 프로젝트 관리 솔루션이 존재한다. 대부분 일정 관리, 업무 배정, 이슈 관리, 협업 기능을 중심으로 발전해 왔다.

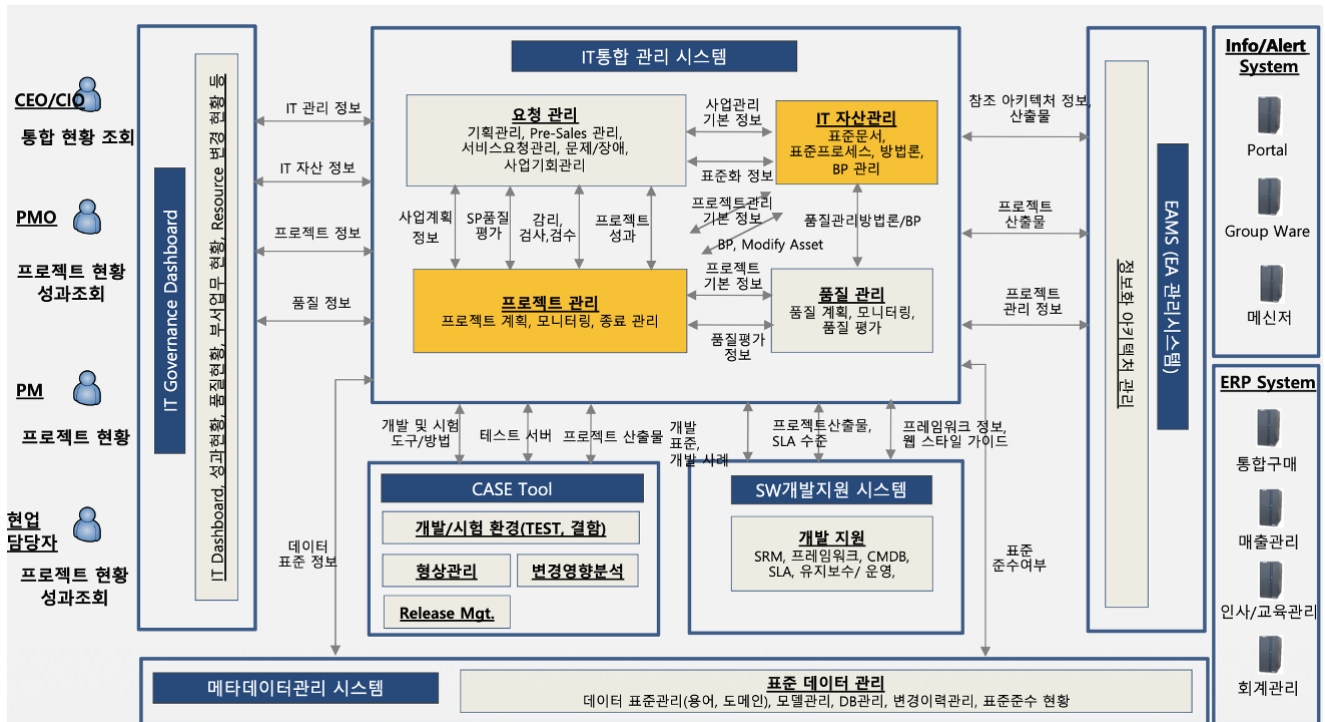
그러나 솔루션링크는 다른 질문을 던졌다.

“프로젝트가 끝난 후 조직에는 무엇이 남는가?”

이 질문이 ITSCOPE를 기존 PMS와 구별하는 가장 큰 차이점이다.

ITSCOPE는 프로젝트 계획부터 요구사항 관리, 일정 관리, 위험 관리, 품질 관리, 형상 관리, 산출물 관리, 의사결정 이력, 협업, 시험과 검증까지 프로젝트 전 생애주기를 하나의 플랫폼에서 관리한다. 그러나 진정한 가치는 관리 기능 자체에 있는 것이 아니다.

프로젝트 수행 과정에서 발생하는 모든 의사결정과 문제 해결 과정, 위험 대응 사례, 품질 개선 경험, 베스트 프랙티스(Best Practice), 교훈(Lessons Learned)을 조직의 자산으로 축적하고, 다음 프로젝트에서 즉시 활용할 수 있도록 설계한 것이 ITSCOPE의 핵심이다.



[ITSCOPE 기능 구성도]

이러한 접근은 AI 시대에 더욱 큰 의미를 가진다. 생성형 AI는 방대한 정보를 기반으로 답변을 생성하지만, 기업의 경쟁력은 공개된 정보가 아니라 조직 내부에 축적된 경험과 노하우에서 나온다. 조직의 지식이 구조화되어 있지 않다면 AI 역시 효과적으로 활용될 수 없다.

결국 AI 시대의 경쟁력은 AI 모델의 크기보다 조직 지식의 품질에 달려 있다.

산업 현장에서 검증된 엔지니어링

솔루션링크의 이러한 철학은 실제 산업 현장에서 검증되어 왔다.

회사는 금융권 차세대 시스템 구축, 공공 정보화 사업, 자동차 전장 소프트웨어, 스마트 제조, 국방 시스템 등 다양한 분야에서 소프트웨어공학 기반의 엔지니어링 서비스를 제공해 왔다. 특히 Automotive SPICE, ISO 26262 기능안전, 시스템 엔지니어링, 프로젝트 품질관리 분야에서는 국내 산업계의 역량 향상에 꾸준히 기여해 왔다.

또한 ITSCOPE는 국내 주요 금융기관을 비롯해 공공기관에서도 활용되고 있으며, 국내 5대 은행 중 4개 은행이 도입한 프로젝트 엔지니어링 플랫폼으로 알려져 있다. 이러한 실적은 프로젝트 관리 기능만이 아니라, 조직의 지식 자산을 체계적으로 관리하고 활용하는 플랫폼으로서의 가치를 인정받은 결과라 할 수 있다.

AI 시대에는 기술보다 경험의 가치가 더욱 커진다. 그리고 그 경험을 조직의 경쟁력으로 전환하는 과정에는 여전히 소프트웨어공학이 중심에 있다.

솔루션링크가 지난 25년 동안 축적해 온 것은 단순한 프로젝트 수행 실적이 아니라, 산업 현장에서 검증된 엔지니어링의 원칙이었다.

AI Engineering과 AI Governance, 그리고 대한민국이 준비해야 할 미래



[민상운 대표 인터뷰 사진]

“AI 시대에도 결국 엔지니어링이 답입니다”

솔루션링크를 이야기할 때 빼놓을 수 없는 인물이 있다. 바로 민상운 대표이다.

그는 기업의 최고경영자이면서 동시에 소프트웨어 공학자이며, AI Engineering과 AI Governance 분야의 전문가로 활동하고 있다. 기업을 운영하는 경영자의 시각과 산업 현장의 엔지니어 경험, 그리고 국제표준 활동을 통해 얻은 글로벌 관점을 함께 갖추고 있다는 점이 그의 가장 큰 특징이다.

최근 AI 열풍 속에서 많은 기업들이 생성형 AI를 도입하는 데 집중하고 있지만, 민 대표가 가장 자주 강조하는 키워드는 의외로 “Engineering”이다.

그는 AI를 단순히 새로운 소프트웨어 기술로 보지 않는다. AI는 앞으로 국가와 산업의 핵심 인프라가 될 것이며, 그릴수록 AI를 어떻게 설계하고, 검증하며, 안전하게 운영할 것인가가 경쟁력을 결정한다고 말한다.

“AI 모델은 계속 발전할 것입니다. 그러나 산업에서 필요한 것은 가장 뛰어난 모델이 아니라 가장 신뢰할 수 있는 시스템입니다. AI는 기술이지만, 신뢰는 공학을 통해 만들어집니다.”

이러한 철학은 솔루션링크가 지난 25년간 축적해 온 소프트웨어공학의 경험과도 자연스럽게 연결된다.

민 대표는 국내 AI 표준화 활동에서도 활발한 역할을 수행하고 있다.

현재 산업표준화포럼 운영위원장과 초대 (사)소프트웨어와사회안전협회 회장을 맡았고, ISO/IEC JTC 1/SC 42를 비롯한 AI 국제표준 활동과 IEEE의 AI 및 소프트웨어 안전 관련 표준화 활동에도 참여하고 있다. 이러한 활동은 단순한 기술 표준 제정을 넘어 우리나라 산업계가 글로벌 AI 시장에서 경쟁력을 확보할 수 있는 기반을 마련하는 데 목적이 있다.

그는 AI 표준을 흔히 생각하는 ‘규제’와는 다른 시각으로 바라본다.

“표준은 혁신을 제한하기 위해 존재하는 것이 아닙니다. 기술이 산업과 사회의 기반이 되는 순간, 지속 가능한 혁신을 가능하게 하기 위해 필요한 약속입니다.”

실제로 산업혁명 이후 등장한 대부분의 핵심 기술은 일정 단계에 이르러 표준과 제도를 통해 산업 전반으로 확산되었다. 철도와 전력, 인터넷, 이동통신이 모두 그랬듯이, AI 역시 산업 인프라가 될수록 국제적으로 통용되는 공통 규범과 기술 체계가 중요해질 수밖에 없다.

특히 생성형 AI와 Physical AI 시대에는 상호운용성(Interoperability), 안전성(Safety), 신뢰성(Reliability), 책임성(Accountability), 보안(Security)과 같은 요소가 제품의 성능만큼 중요한 경쟁력이 될 것으로 전망된다.

Physical AI 시대, 새로운 도전

최근 AI 산업의 화두는 Physical AI다.

기존 생성형 AI가 문서와 이미지, 음성을 처리하는 디지털 공간에서 활동했다면, Physical AI는 로봇과 자율주행차, 스마트 제조설비처럼 현실 세계에서 직접 사람과 상호작용한다.

이러한 환경에서는 AI의 정확도만으로는 충분하지 않다.

예를 들어 휴머노이드 로봇이 공장에서 작업을 수행한다고 가정해 보자. 각각의 AI 기능이 95~99%의 성공률을 보이더라도, 수십 개의 작업이 연속적으로 이루어지는 실제 제조공정에서는 전체 시스템의 신뢰성이 급격히 낮아질 수 있다.

따라서 Physical AI 시대에는 개별 AI 모델보다 시스템 전체의 안전성과 신뢰성을 확보하는 기술이 더욱 중요해질 것으로 예상된다.

민 대표는 이를 “AI Safety를 넘어 Digital Safety Engineering이 필요한 시대”라고 설명한다.

대한민국 AI 산업이 준비해야 할 것

인터뷰 말미에 “대한민국 AI 산업이 앞으로 가장 먼저 준비해야 할 것이 무엇이라고 생각하십니까?”라는 질문을 던졌다.

민 대표의 대답은 의외로 간단했다.

“AI를 잘 만드는 것도 중요하지만, AI를 잘 사용하는 나라가 되어야 합니다.”

그는 AI 시대의 경쟁력을 다음과 같은 세 가지 요소로 설명했다.

첫째, 산업 현장에서 활용할 수 있는 AI Engineering 역량.

둘째, 글로벌 시장과 연결되는 국제표준과 AI Governance 역량.

셋째, 조직의 경험을 지속적으로 축적하고 발전시키는 Engineering Knowledge.

이 세 가지가 함께 성장해야만 대한민국이 AI를 소비하는 국가가 아니라 AI 산업을 선도하는 국가가 될 수 있다는 것이다.

맺음말

이번 기업탐방을 준비하며 가장 인상 깊었던 것은 솔루션링크가 AI를 이야기하면서도 끝까지 소프트웨어공학을 놓지 않았다는 점이다.

생성형 AI는 개발 방식을 바꾸고 있다.

Physical AI는 산업의 모습을 바꾸고 있다.

AI Governance는 국제 질서를 바꾸고 있다.

그러나 그 모든 변화의 중심에는 여전히 요구사항을 정의하고, 시스템을 설계하며, 품질과 안전성을 확보하는 공학적 사고가 존재한다.

25년 전 소프트웨어공학의 중요성을 이야기하던 솔루션링크는 이제 AI Engineering과 Digital Safety Engineering이라는 새로운 언어로 같은 철학을 이어가고 있다.

“Safer Way to Smarter Future.”

이 슬로건은 단순한 기업의 비전이 아니다. AI가 산업과 사회의 핵심 인프라가 되어가는 시대에, 기술과 혁신을 신뢰로 연결하기 위한 소프트웨어공학의 새로운 역할을 상징하는 메시지가기도 하다.

AI 시대가 열릴수록 소프트웨어공학은 사라지는 것이 아니라 오히려 그 가치가 더욱 커질 것이다. 그리고 그 변화의 한가운데에서 솔루션링크는 산업과 공학, 그리고 국제표준을 연결하는 새로운 길을 만들어가고 있다.

Editor's Note

AI는 더 이상 미래 기술이 아니라 산업과 사회를 움직이는 핵심 인프라가 되고 있다. 기술 혁신의 속도가 빨라질수록 이를 안전하고 지속 가능하게 활용하기 위한 소프트웨어공학의 역할은 더욱 중요해질 것이다. 이번 기업탐방은 이러한 변화 속에서 산업 현장과 국제표준을 연결하며 AI Engineering의 새로운 방향을 제시하고 있는 솔루션링크의 사례를 통해, AI 시대 소프트웨어공학의 미래를 함께 생각해 보는 계기가 되기를 기대한다.



소사이어티 광장

축하합니다!

• 수여

- 박사학위 수여: KAIST 김태현 2026년 8월 (지도교수: KAIST 백종문)

• 수상

- KAIST COINSE 연구실 FSE 2026 ACM Distinguished Paper Award 수상

• KAIST 계산지능형소프트웨어공학 연구실 (COINSE) 윤주연 (박사과정), 김소민 (박사과정), 유신 (지도교수) 연구팀이 스웨덴 Chalmers University의 Robert Feldt 교수와 함께 소프트웨어공학 최우수학술대회인 Foundations of Software Engineering (FSE 2026)에서 ACM SIGSOFT Distinguished Paper Award를 수상하였다. 연구팀이 개발한 'Clotho'는 특정 프롬프트를 입력했을 때 LLM이 생성할 최종 응답이 올바른지를 출력 토큰 생성 전에 예측하는 기술이다. 응답 생성 전 검증을 통해 LLM을 평가하는 데 필요한 시간과 비용을 줄일 수 있으며, 총 211편의 발표 논문 중 10편의 수상작에 선정돼 우수성을 인정받았다.

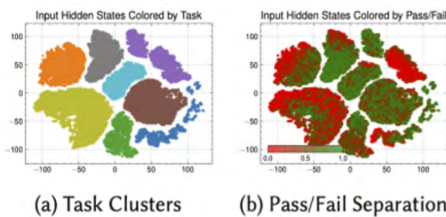


Fig. 1. Visualisation on Llama 8B internal state space using inputs for all studied tasks



소개합니다!

• 책 출간



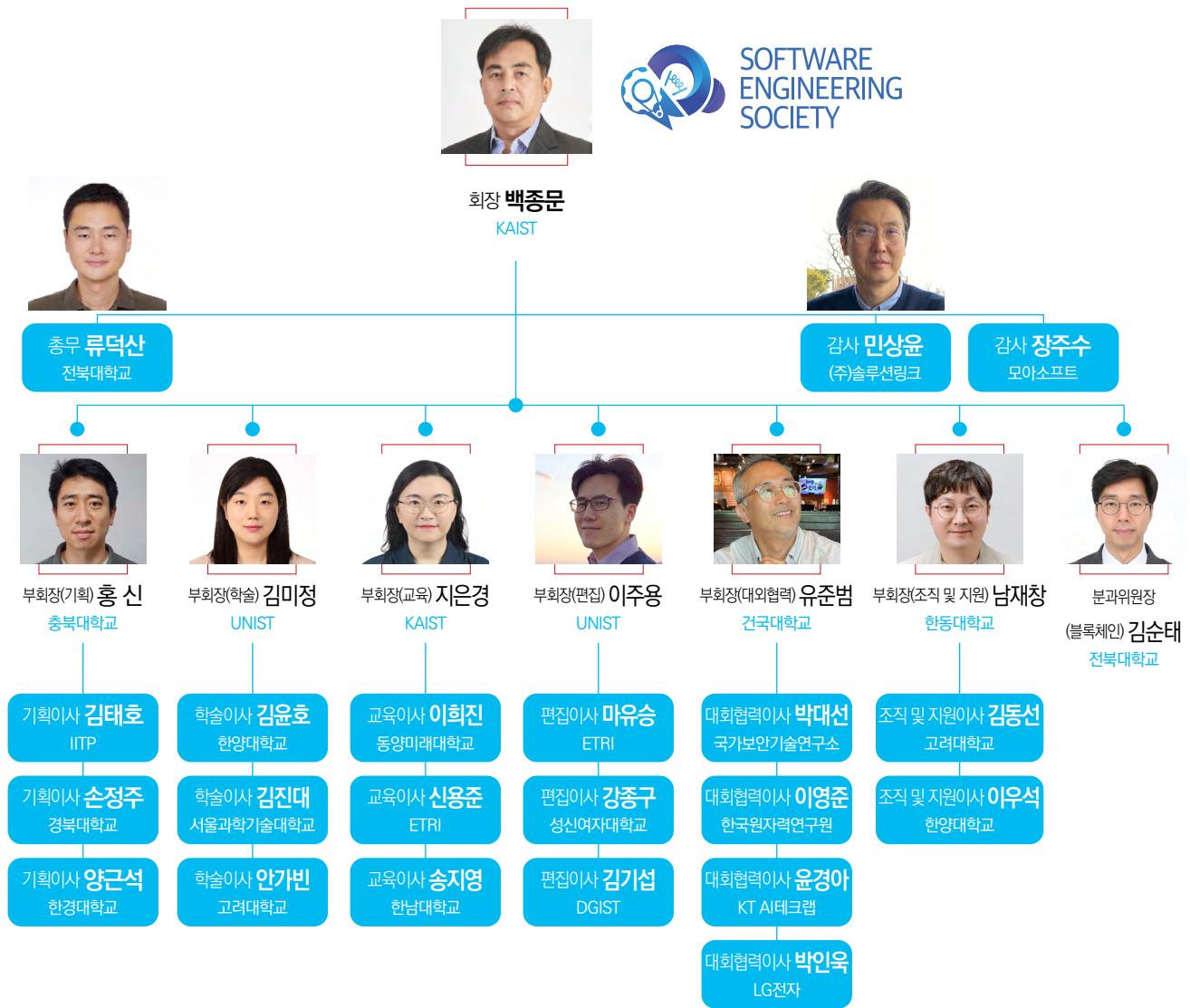
- KAIST 전산학부 강성원 명예교수가 『실용 논리학: 논증의 분석과 평가(황소걸음)』를 집필하여 2026년 3월 출간하였습니다.

- 일상에서 쓰는 논증은 대부분 비연역적이다. 연역적으로 타당하지 않은 논증을 배제하면 실용적 판단은 불가능하다. 비연역적이면서도 합리적인 논증의 광활한 세계가 있다. 실용 논리학은 독자에게 그 세계를 소개하고 또한 그 세계를 이해하고 파헤칠 수 있는 도구를 제공한다. 대립되는 정치적 발언이 난무하는 가운데 어느 쪽 말을 받아들여야 할지 어떻게 판단할 수 있을까? 실용 논리학은 간단한 일상의 문제에서부터 복잡한 정치 사회 문제에 이르기까지 우리 주위에서 발생하는 쟁점들의 해결에 필요한 원리들을 소개한다.

- 강성원 명예교수의 저서로는 『소프트웨어 아키텍처로의 초대(2012)』, 『체계적인 소프트웨어 제품라인 개발(2017)』, 『연역 논리학: 논증의 증명과 반박(2025)』, 공동 저서로는 『소프트웨어 제품라인 개발 입문(2021, 이지현, 한영훈, 정필수 공저)』, 번역서로는 『지식인의 자격(2024, 윤종은 공역)』이 있다.



: 소사이어티 조직도



발행정보

발행일 2026년 9월 1일

발행인 백종문

발행처 사단법인 한국정보과학회 소프트웨어공학소사이어티

연락처 대전광역시 유성구 대학로 291, N1빌딩 502호 (우: 34141)

백종문

홈페이지 <https://www.sigsoft.or.kr>





사단법인 한국정보과학회 소프트웨어공학소사이어티

주소: 대전광역시 유성구 대학로 291, N1빌딩 502호 (우: 34141)
백종문

홈페이지: <https://www.sigsoft.or.kr>