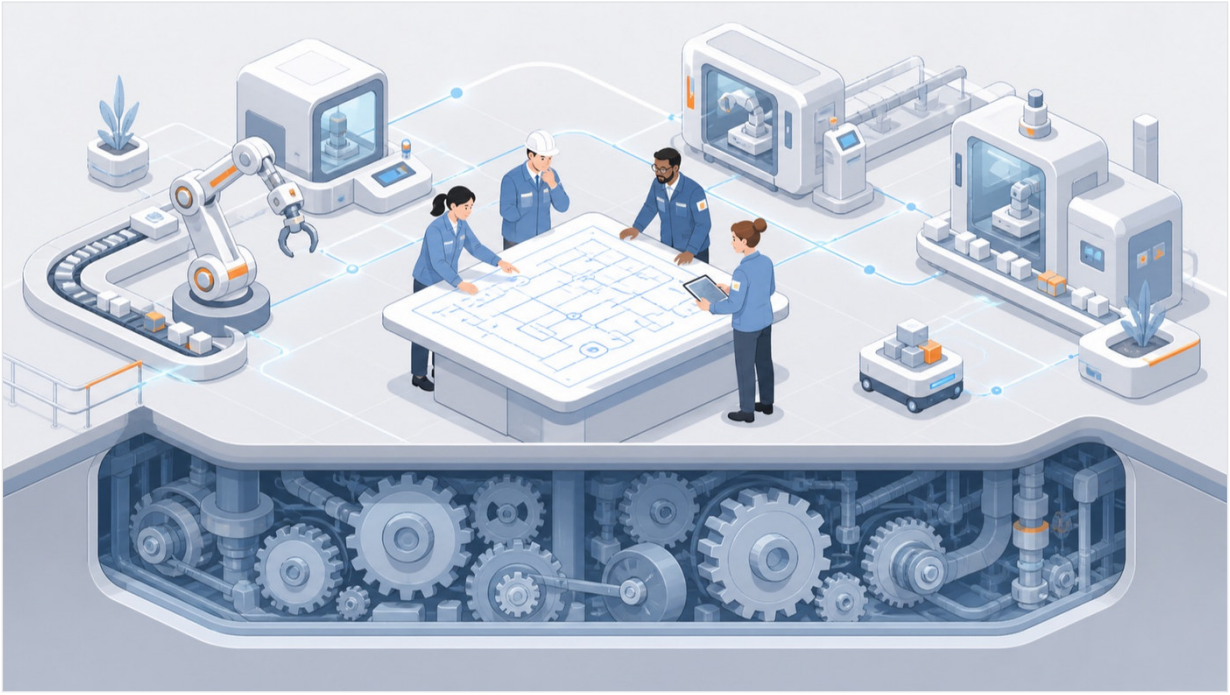
인사이트

저희가 바이트코드를 안 읽어도 됐던 이유

방동걸 (주)웨이스 대표이사
2026년 8월 15일

며칠 전 「코드를 모르는 개발자의 시대가 온다」는 칼럼을 읽었습니다. 개발자 닉 하지스(Nick Hodges)가 ITWorld에 쓴 글입니다. 논지는 이렇습니다. 아무도 JVM 바이트코드나 .NET CIL을 손으로 쓰지 않는 것처럼, 머지않아 타입스크립트와 파이썬도 손으로 쓰지 않게 된다. 그러면 사람이 읽을 수 있는 중간 언어가 필요할 이유가 없어지고, 에이전트가 자기에게 맞는 언어를 스스로 만들 것이다. 코드의 척도는 "작동하느냐, 그렇지 않느냐"로 단순해지고, 사람은 입력만 담당한다. 웨이스가 그 논지를 매일 VEXPLOR 코드를 쓰는 회사 기준으로 다시 읽어 보았습니다.

절반은 맞는 이야기라고 생각합니다. 다만 이 논지에는 항이 하나 빠져 있고, 그 빠진 항이 제조업에서는 결론을 통째로 뒤집습니다.



바이트코드를 안 읽어도 됐던 것은 아래층이 확정적이었기 때문입니다.

1. 빠진 항은 추상화가 아니라 확정성입니다

우리가 바이트코드를 안 읽어도 됐던 이유는 그것이 기계용 언어라서가 아닙니다. **컴파일러가 확정적이기 때문**입니다. 같은 소스는 언제나 같은 바이트코드를 만듭니다. 그래서 소스 수준에서 확인하면 그 아래가 자동으로 보증됐고, 우리는 검사할 위치를 한 계단 위로 올릴 수 있었습니다.

즉 컴파일러가 없앤 것은 중간 표현이 아니라 **중간 표현을 따로 검사할 필요**였습니다. 그리고 그 필요를 없애준 것은 추상화가 아니라 확정성입니다.

언어 모델은 확정적이지 않습니다. 같은 지시가 호출마다 기능적으로 다른 코드를 만들고, 명세에 비어 있던 부분은 재생성할 때마다 다른 모습으로 되살아납니다. 위에서 확인하면 아래가 보증된다는 관계가 성립하지 않습니다. 그러면 중간 표현을 지울 근거가 생기는 게 아니라, 확인할 일이 오히려 늘어납니다.

한 문장으로 줄이면 이렇습니다. **컴파일러는 검증을 없앴고, 에이전트는 검증을 옮겼을 뿐입니다.**

2. "되면 된 거지"는 답이 아니라 미뤄둔 질문입니다

"작동하느냐로 단순해진다"는 문장은 작동 여부를 판정하는 일이 공짜라고 가정합니다. 실제로는 그 판정이 가장 비쌉니다.

코딩 에이전트 평가에 널리 쓰이던 SWE-bench Verified를 두고 METR이 내놓은 후속 분석에 따르면, 테스트를 통과한 에이전트 제출물 가운데 **절반가량은 실제 관리자라면 반영하지 않을 것**으로 판정됐습니다. OpenAI 역시 내부 감사에서 문제 항목 상당수가 시험 자체의 결함으로 애초에 풀 수 없는 문제였음을 확인하고 이 지표 보고를 중단했습니다. 테스트를 통과했다는 것과 작동한다는 것이 같은 말이 아니었던 겁니다.

현장의 체감도 같은 방향입니다. 스택 오버플로 2025년 개발자 조사에서 AI 도구를 쓰거나 쓸 계획인 개발자는 84%였지만 정확도를 신뢰한다는 응답은 29%로 떨어졌고, 가장 큰 불만으로 45%가 꼽은 것은 **"거의 맞는데 정확히는 아닌"** 결과물이었습니다. 응답자의 66%는 그 "거의 맞는" 코드를 고치는 데 시간을 더 쓴다고 답했습니다.

작동한다는 판정은 명세 대비로만 내려집니다. 명세가 자연어면 판정도 자연어의 모호함을 그대로 물려받습니다. 소프트웨어가 오랫동안 코드를 붙잡아 온 이유는 코드가 사람용 표현이어서가 아니라, 기계가 검사할 수 있는 유일한 명세였기 때문입니다.



사야 할 것은 속도가 아니라 근거입니다.

---

3. 코드는 쏟아지는데 이해는 늘지 않습니다

지난 5년간의 커밋 2억 1,100만 줄을 분석한 깃클리어(GitClear) 조사는 방향이 뚜렷합니다. 5줄 이상 중복 블록이 2024년 한 해에 8배로 늘었고, **복사·붙여넣기가 사상 처음으로 코드 이동(리팩터링)을 넘어섰습니다.** 리팩터링에 해당하는 변경 비중은 2021년 25%에서 2024년 10% 아래로 내려갔습니다.

구글이 발표한 2025년 DORA 보고서는 두 결과를 나란히 놓습니다. AI 도입은 처리량과 뚜렷하게 정비례하지만, **동시에 불안정성과도 정비례합니다.** 변경 실패와 재작업이 늘고, 테스트·리뷰·품질 확인이라는 하류 구간이 병목으로 드러납니다. 그리고 이 보고서에서 가장 서늘한 문장은 이것입니다. **도입 전에 성숙했던 조직도 이 품질 저하에서 예외가 아니었습니다.**

피터 나우르가 1985년에 쓴 표현을 빌리면, 프로그램은 소스코드가 아니라 만든 사람들이 가진 "왜 이렇게 만들었는가"의 이론이고 코드는 그 이론의 손실 압축입니다. 작성을 자동화하면 코드는 늘지만 이론은 늘지 않습니다. 지금 측정되는 유지보수성 악화는 그 격차입니다.

4. 제조에서는 이 논쟁이 성립하지 않습니다

여기까지는 소프트웨어 업계의 이야기입니다. 저희가 일하는 제조 현장으로 오면 전제 자체가 달라집니다.

기능안전 표준 IEC 61508은 **요구사항과 코드와 시험 사이의 양방향 추적성**을 명시적으로 요구합니다. 모든 요구사항이 빠짐없이 구현됐다는 것과, 어떤 요구사항에도 대응하지 않는 잉여 코드가 없다는 것을 함께 증명해야 합니다. 산업 보안 표준 IEC 62443도, 제약·화장품 계열의 GAMP 5도 같은 계열의 요건을 겁니다.

여기서 요구되는 것은 "돌아간다"가 아닙니다. **"왜 이 코드가 존재하는지를 증거로 제시할 수 있다"** 입니다. 그러니 "척도는 작동하느냐뿐"인 세계는 제조에 제도적으로 존재할 수 없습니다. 설비와 사람이 걸려 있고, 그 위에 인증이 걸려 있기 때문입니다.

그래서 제조에서 AI는 이런 모습으로 작동합니다. 작성을 아무리 가속해도, 원래도 가장 비쌌던 검증과 적합성 입증 쪽으로 압력이 그대로 몰립니다. 빨라진 만큼 증명할 일이 늘어납니다.

05

사야 할 것은 속도가 아니라 근거입니다

저희가 공장 자율화를 설계할 때 자율의 크기를 증거의 크기에 맞춰 늘리는 것도 같은 이유입니다. AI가 판단의 근거 데이터를 먼저 내놓고, 사람이 그 근거를 보고 승인 범위를 넓혀 갑니다. 무엇을 왜 그렇게 판단했는지가 기록으로 남지 않으면, 그 판단은 공장에서 쓸 수 없습니다.

칼럼의 마지막 우려에 대해서도 한마디 덧붙이고 싶습니다. 주니어 개발자가 필요 없어지면 시니어도 필요 없어질 것이라는 전망인데, 스탠퍼드 디지털이코노미랩이 미국 급여 데이터로 분석한 결과는 방향이 다릅니다. AI 노출이 큰 직군에서 22~25세 초기경력 고용은 뚜렷하게 줄었지만 **경력자 고용은 안정적으로 유지**됐고, 감소는 AI를 자동화에 쓴 경우에만 나타났습니다. 판정할 사람은 덜 필요해진 게 아니라 더 필요해졌습니다. 다만 그 사람을 길러내던 경로가 얇아지고 있을 뿐입니다.

그러니 지금 도입을 검토하신다면, 질문은 "얼마나 빨리 만들어 줍니까"가 아니라 이쪽이어야 합니다.

"그렇게 만든 것을, 무엇으로 증명해 주십니까?"

코드를 모르는 개발자의 시대는 오지 않습니다. 코드를 쓰지 않고 **코드를 판정하는** 개발자의 시대가 올 뿐입니다. 그리고 제조에서는 그 판정이 곧 인증이고, 인증이 곧 사업입니다.

웨이스가 하는 일이 궁금하시면: wace.me

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문 blog.vexplor.com/post/why-we-never-read-bytecode
발행 2026년 8월 15일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인