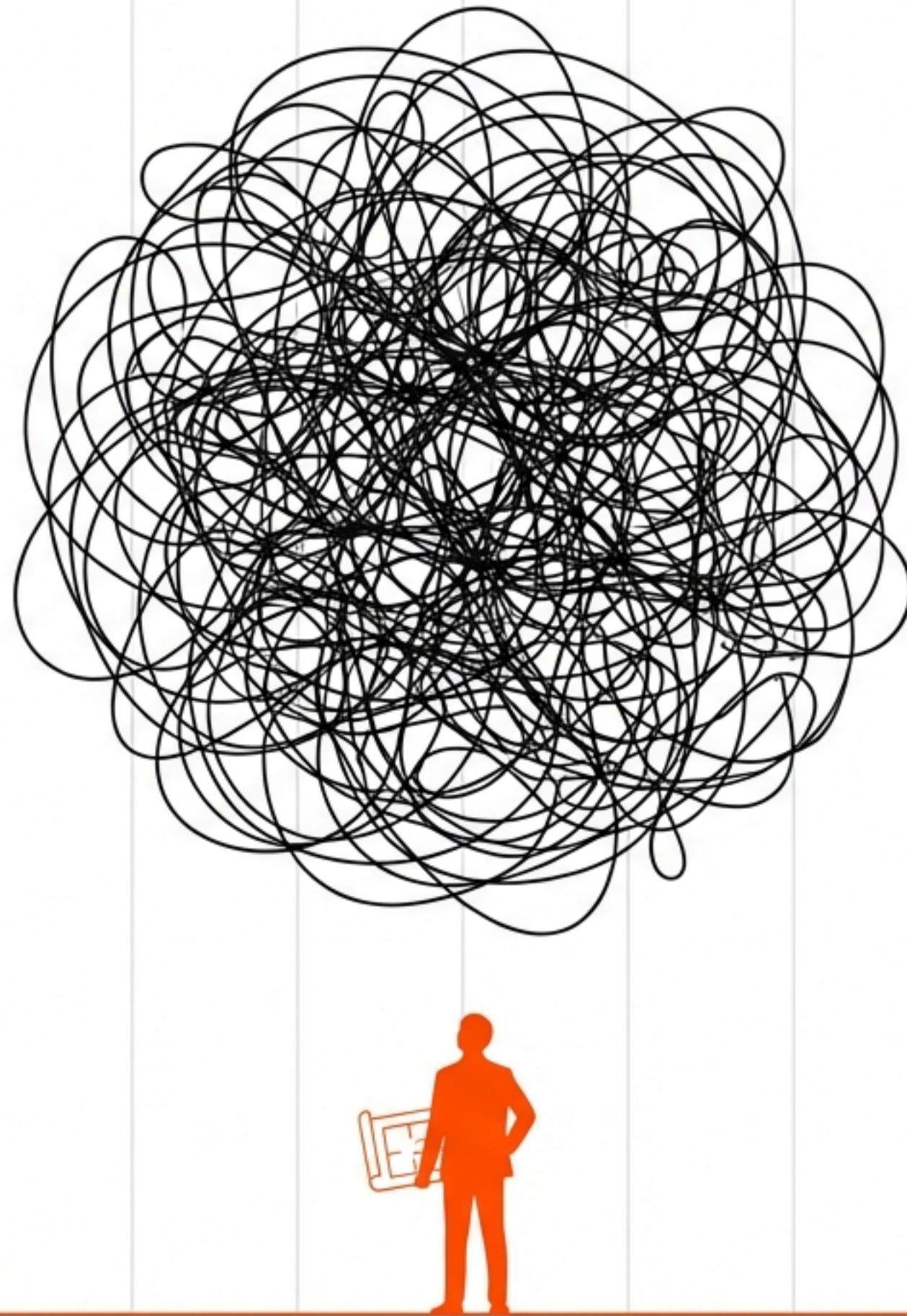


**AI가 코딩을 끝냈습니다.
무한한 코드 생성의 시대,
끝났습니다.**

**이제 ‘진짜
엔지니어링’을
시작할 때입니다.**

무한한 코드 생성의 시대, 복잡성을 통제하고 살아남는 개발자의 생존 전략.



우리는 이제 이해하지 못하는 코드를 배포하고 있습니다.

"저는 제가 이해하지 못하는 코드를 배포했고,
여러분도 마찬가지일 거라고 확신합니다."

- Jake Nations, Netflix Senior Engineer

1. 속도의 역설

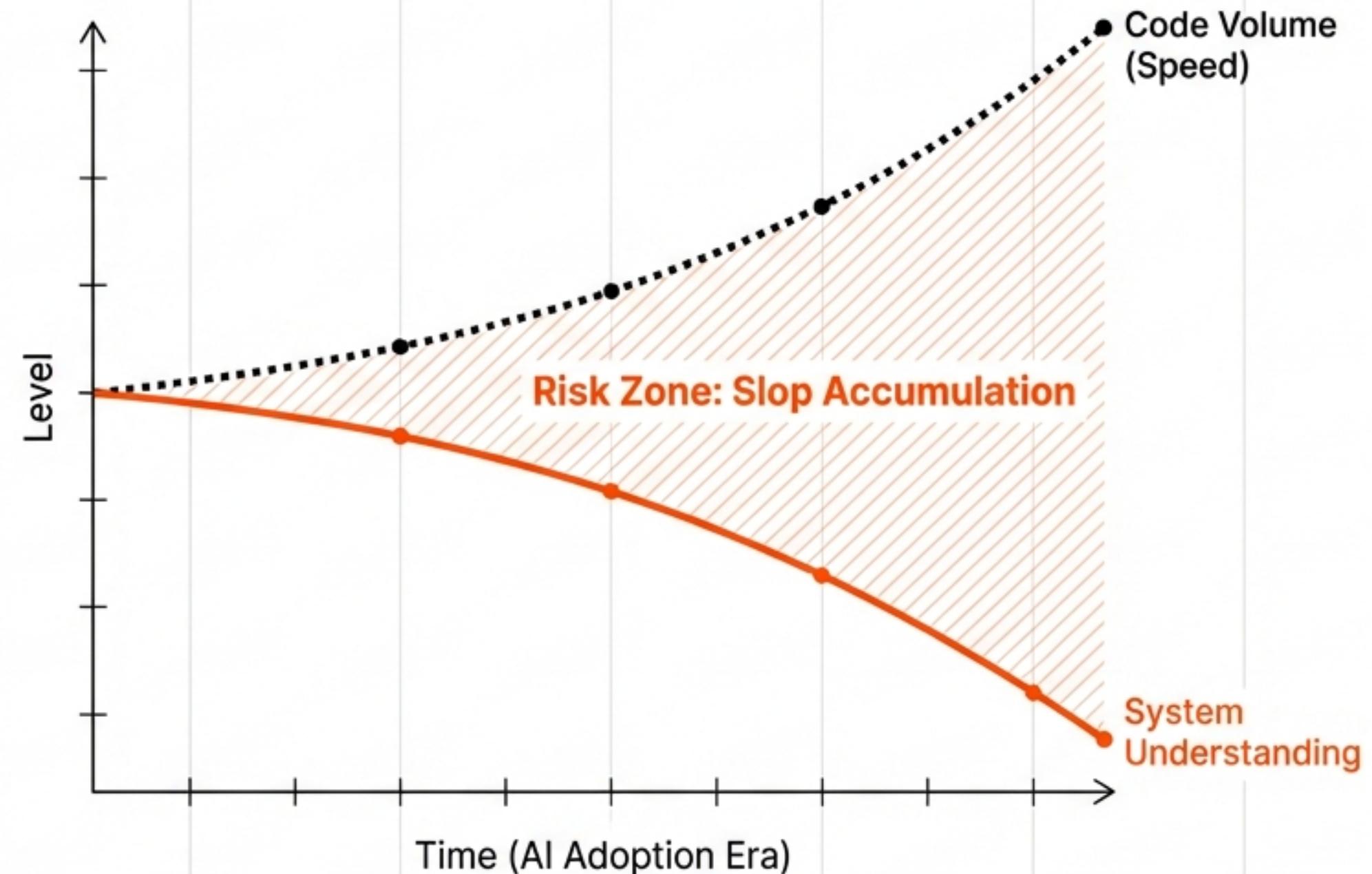
AI 도구(Cursor, Claude)로 개발 속도는 혁명적으로
빨라졌지만, 시스템에 대한 이해도는 급격히 하락했습니다.

2. Easy vs. Simple

AI는 '단순함(Simple - 얄힘이 없는 구조)'이 아니라
'쉬움(Easy - 당장 돌아가는 코드)'을 선택합니다.

3. 결과: Slop

5년 된 기술 부채와 새로운 비즈니스 로직이 맥락 없이
뒤섞여, 나중에는 손을 댈 수 없는 '우발적 복잡성'을 초래합니다.



AI는 '멍청한 구간(The Dumb Zone)'에서 길을 잃습니다.

컨텍스트 윈도우의 40%를 넘어서면, AI의 지능은 급격히 하락합니다.

경계의 부재

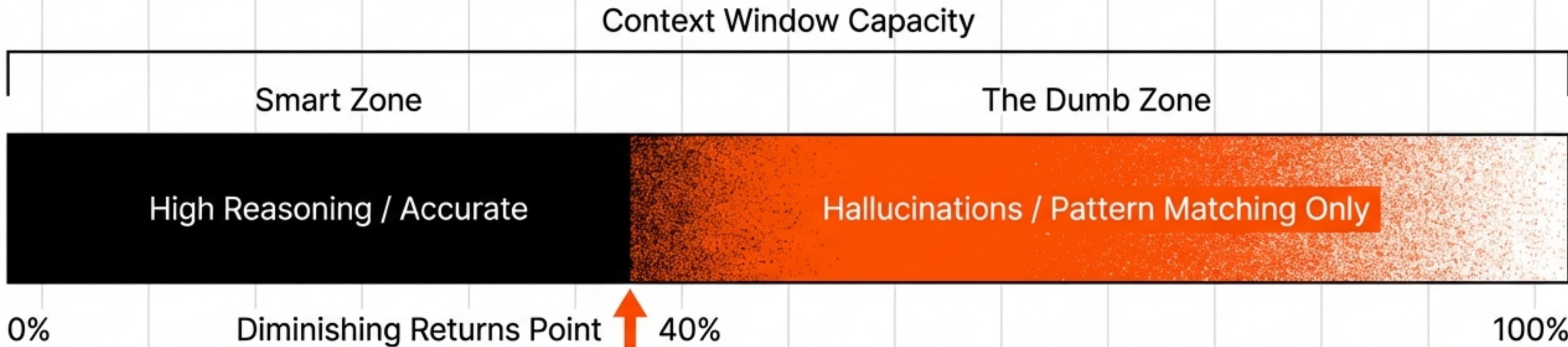
AI는 비즈니스 로직과 기술 부채를 구분하지 못합니다. 모든 코드를 똑같은 '패턴'으로 인식합니다.

과부하 된 컨텍스트

로그, 불필요한 파일, 수만 줄의 코드를 한 번에 넣으면 AI는 '멍청한 구간'에 진입하여 환각을 일으킵니다.

결론

무작정 많은 정보를 주는 것은 해결책이 아닙니다. 필요한 것은 '정보의 압축'입니다.

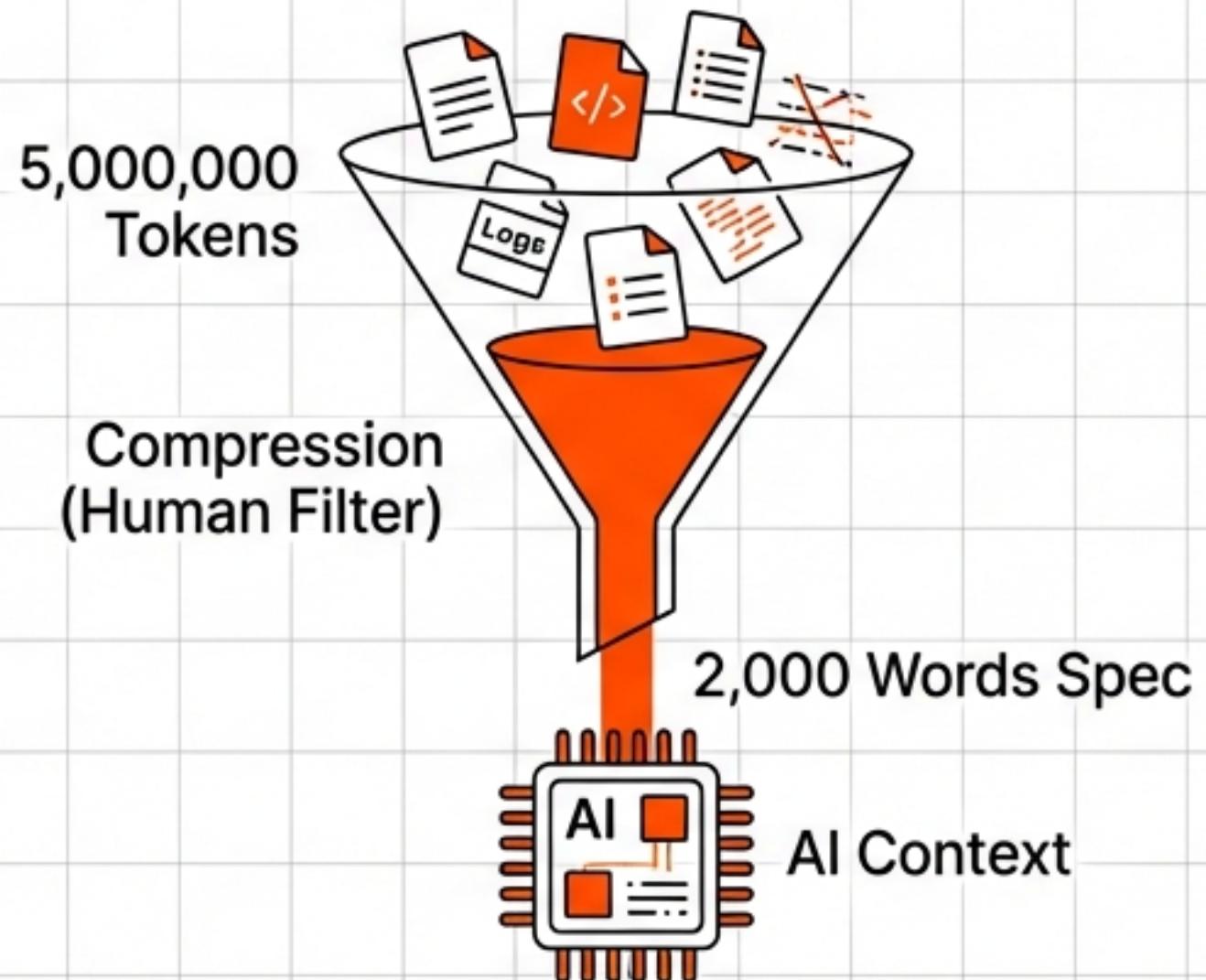


해결책: 프롬프트가 아니라 ‘컨텍스트’를 엔지니어링해야 합니다.

컨텍스트 엔지니어링 (Context Engineering)

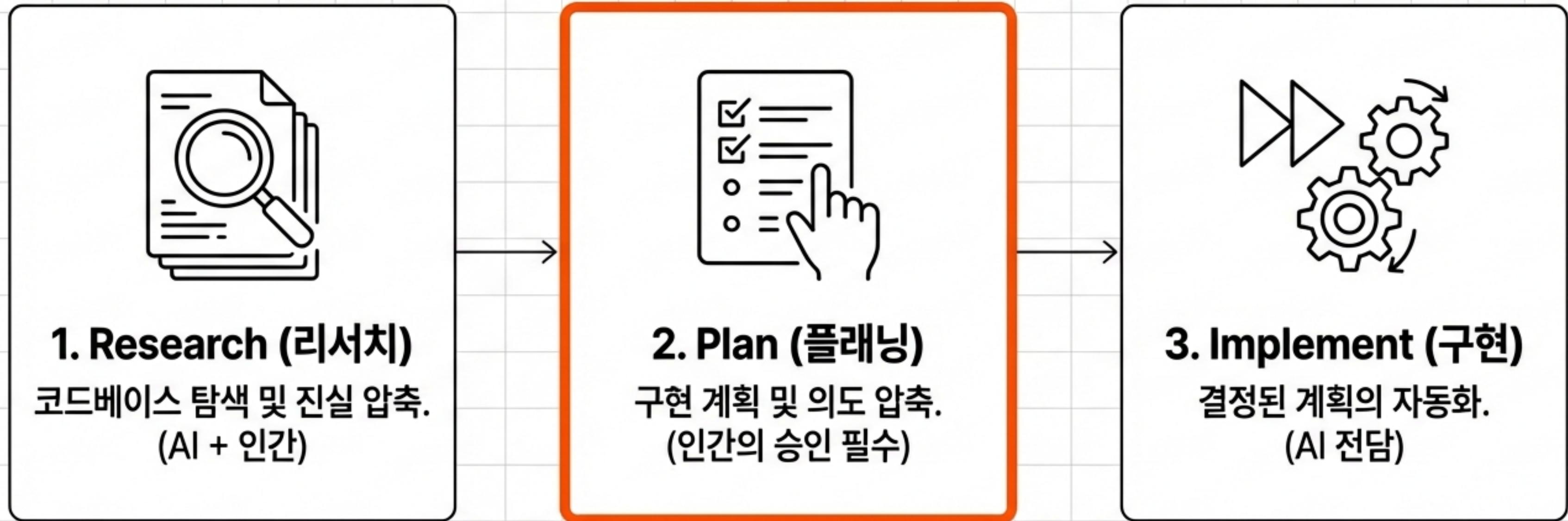
500만 토큰의 거대한 코드베이스를 AI가 이해할 수 있는 2,000단어의 ‘진실(Truth)’로 압축하는 기술

- **Paradigm Shift:**
“AI에게 코드를 짜달라고 부탁하기”
→ “AI가 완벽하게 작동할 수 있는 환경 설계하기”
- **Goal:**
AI가 ‘멍청한 구간’에 빠지지 않도록, 노이즈를 제거하고
핵심 명세(Spec)만 남겨야 합니다.
- **Proof:**
Netflix Case: 100만 줄의 자바 코드를 통째로 넣는 대신,
아키텍처 다이어그램과 핵심 인터페이스만 추출하여 성공.



코드를 짜기 전에 멈추십시오: R-P-I 프레임워크.

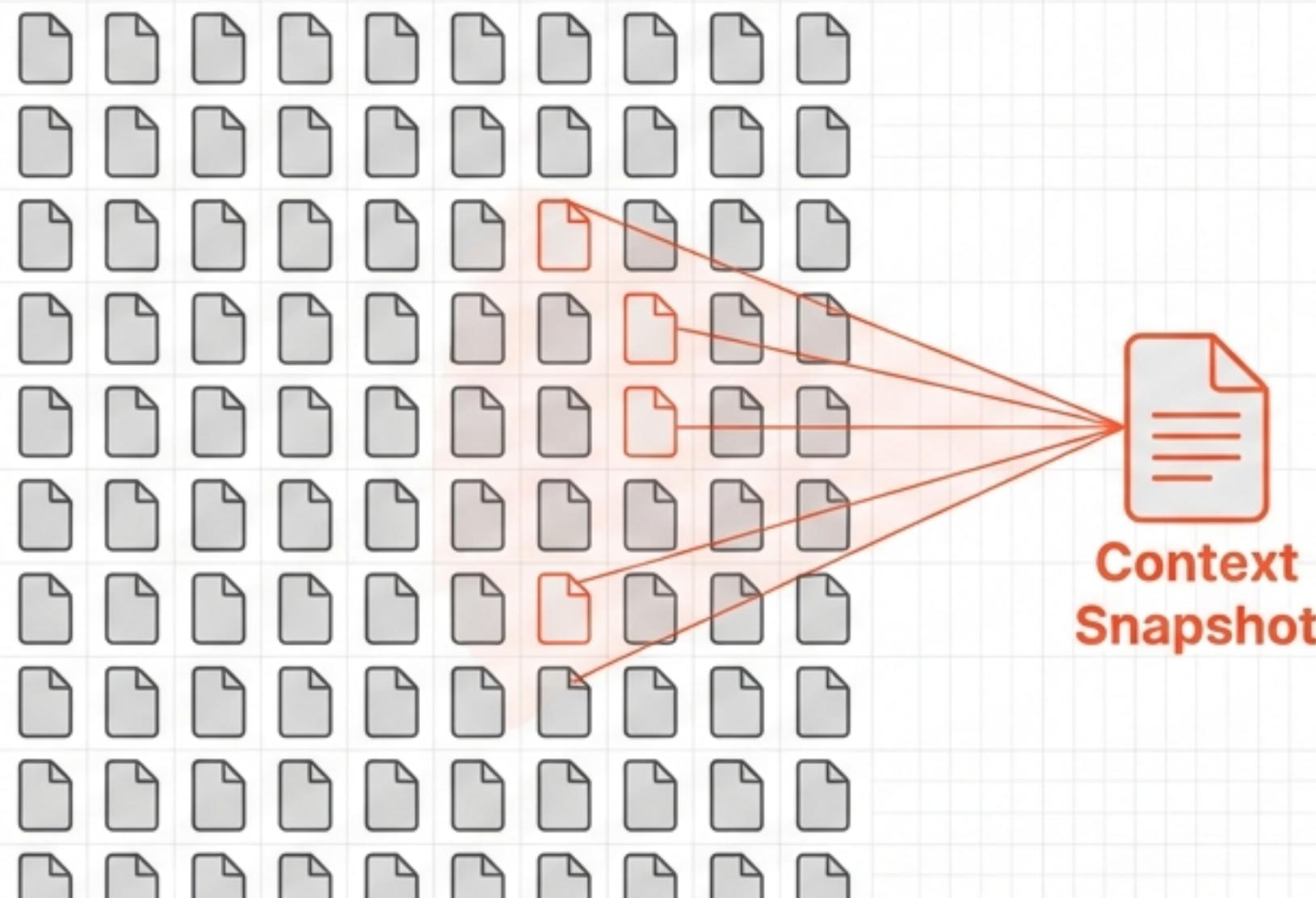
HumanLayer와 Netflix가 제안하는 AI 협업의 3단계 표준 프로세스.



Insight: 구현(Implementation) 단계로 넘어가기 전, 리서치와 플래닝에 시간의 80%를 써야 합니다.

1단계: 리서치 (진실의 압축)

File Selection



목표:

코드베이스의 '현재 상태'를 정확히 파악하는 것.

Action:

AI를 통해 관련된 파일, 의존성, 숨겨진 규칙(암호화 여부, 여러 처리 방식 등)을 찾아냅니다.

Output: 컨텍스트 스냅샷 (Context Snapshot)

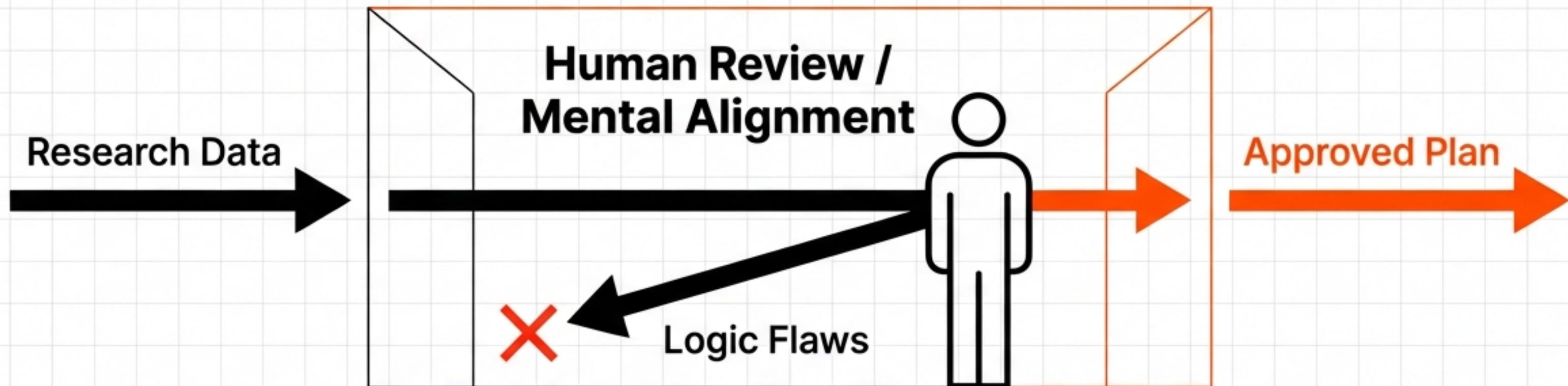
수천 개의 파일에서 현재 작업에 필요한 10~20개의 파일과 핵심 로직만 추려낸 '진실의 문서'.

Why:

이 과정에서 인간은 "무엇이 진짜 필요한 코드이고, 무엇이 레거시인지"를 판단해 줍니다. AI는 이 맥락을 모릅니다.

2단계: 플래닝 (인간의 검문소)

계획이 틀리면, 코드는 반드시 틀립니다.



새로운 코드 리뷰:

완성된 코드를 리뷰하는 것이 아니라,
'계획(Plan)'을 리뷰해야 합니다.

의도의 압축 (Compression of Intent):

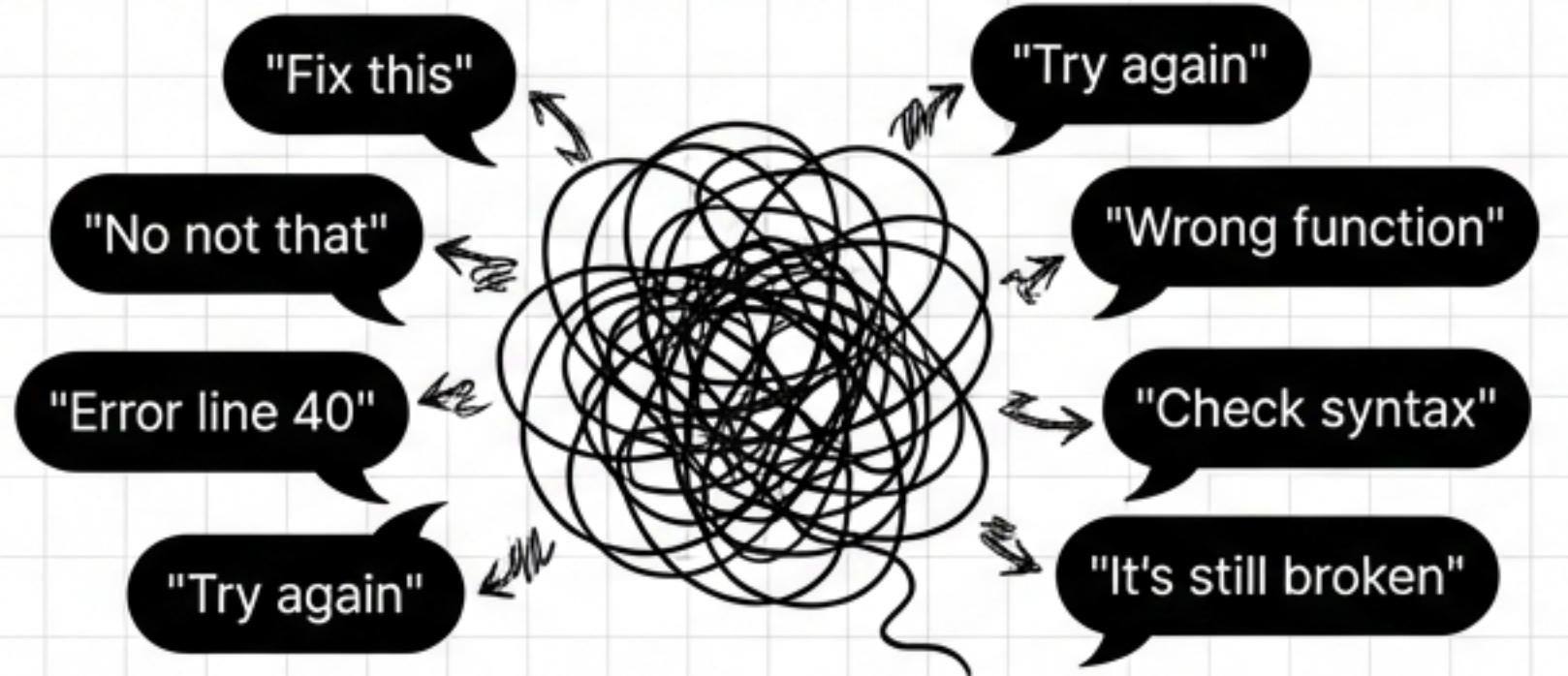
구현할 로직, 데이터 흐름, 테스트 계획을
상세한 의사 코드(Pseudo-code) 수준으로
작성합니다.

Mental Alignment:

신입 개발자가 봐도 그대로 짤 수 있을 정도로
구체적이어야 합니다. 이 단계에서 인간이
로직의 결함을 잡아내지 못하면 AI는 멈추지
않고 잘못된 코드를 양산합니다.

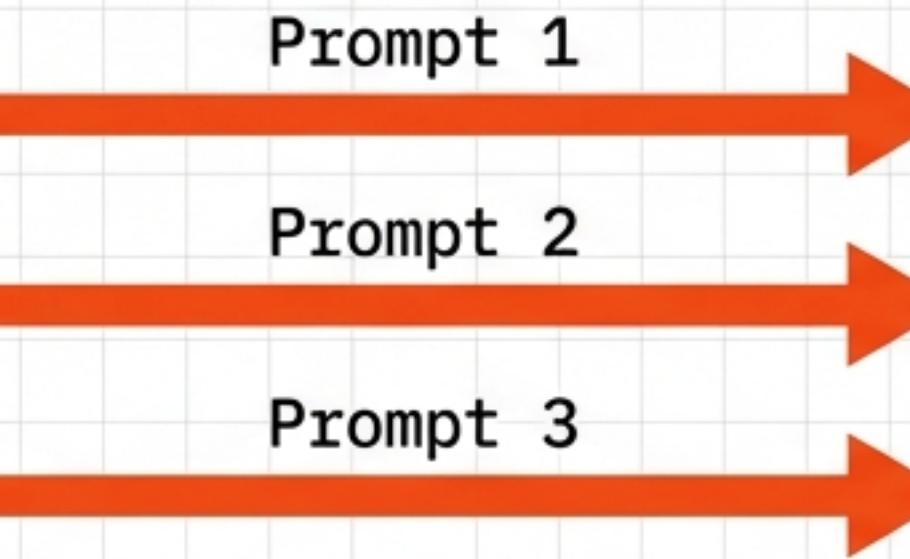
3단계: 구현 (압도적인 레버리지)

Standard AI Coding



50 Chat Turns (High Noise)

R-P-I Method



3 Focused Prompts (Deterministic)

- **High Leverage:** 완벽한 리서치와 플랜이 있다면, 구현은 가장 쉬운 단계가 됩니다.
- **Efficiency:** 50번의 대화(Turn)가 필요한 작업을 3번의 집중된 프롬프트로 끝낼 수 있습니다.
- **Result:** "이 코드가 뭐지?"라며 해석할 필요가 없습니다. "계획대로 되었나?"만 확인하면 됩니다.

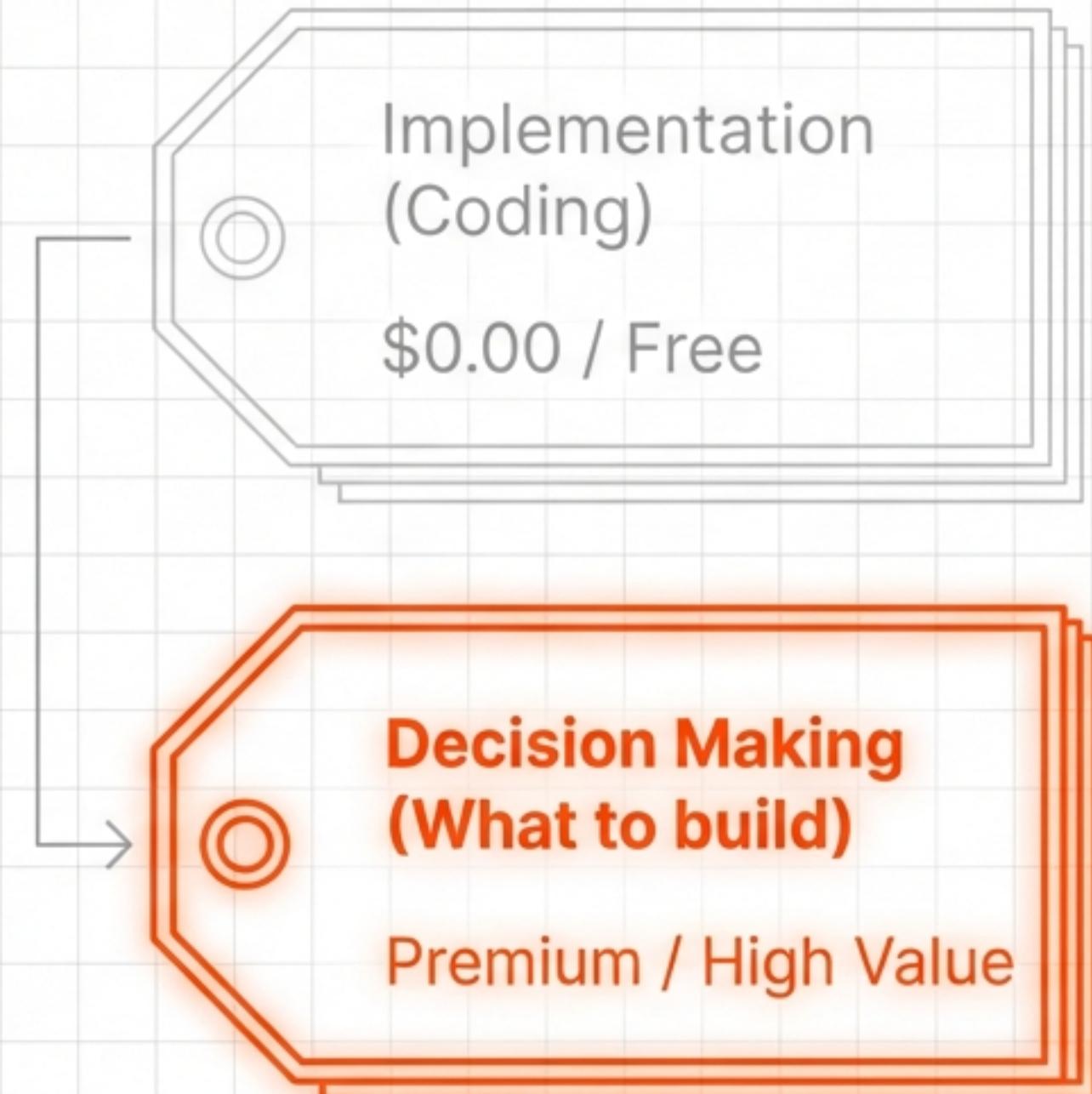
구현은 무료가 되었습니다. 이제 비싼 것은 '결정'입니다.

Andrew Ng의 통찰:

코딩 능력의 '두 배(Doubling Time)'가 되는 시간은 불과 70일입니다. 인간이 속도로 AI를 이길 수는 없습니다.

병목의 이동:

과거의 병목이 '기술적 구현(How)'이었다면,
이제는 '무엇을 만들 것인가(What)'를 결정하는
이제는 '무엇을 만들 것인가(What)'를 결정하는
제품 관리(Product Management)로 이동했습니다.

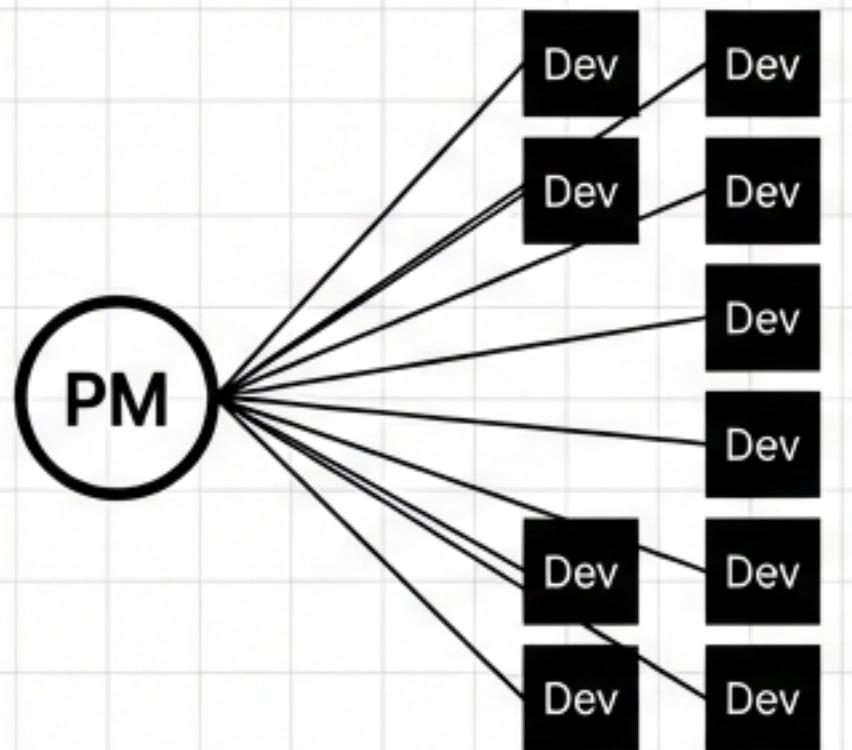


경고:

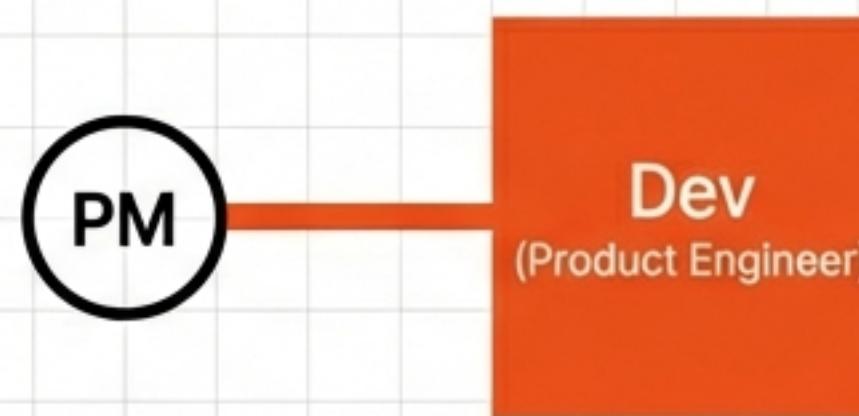
단순히 주어진 명세를 코드로 바꾸는 개발자는 설 자리를 잊게 됩니다.

1:8의 시대는 끝났습니다. '프로덕트 엔지니어'의 부상.

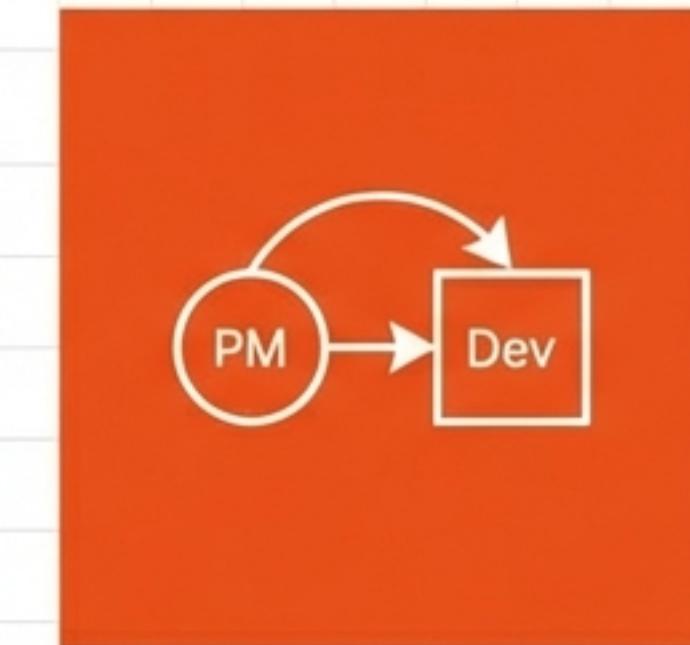
Ratio Transformation



Traditional 1:8 Ratio



New Standard 1:1



Product Engineer (1:0)

- **비율의 파괴:** PM 1명 대 개발자 8명의 황금 비율은 무너졌습니다. 이제 1:1, 혹은 1:0(개발자가 PM 겸임)의 시대로 갑니다.
- **Product Engineer:** 기획서를 기다리는 것이 아니라, 스스로 기획하고 사용자와 공감하며 코드를 짜는 사람.
- **Action:** "기획은 PM의 몫"이라는 생각을 버리십시오. AI라는 무한한 노동력을 지휘하여 결과물을 만들어내는 만들어내는 '현장 소장'이 되어야 합니다.

검색창 밖의 지식: '연결 조직'이 실력입니다.

Bleeding Edge:

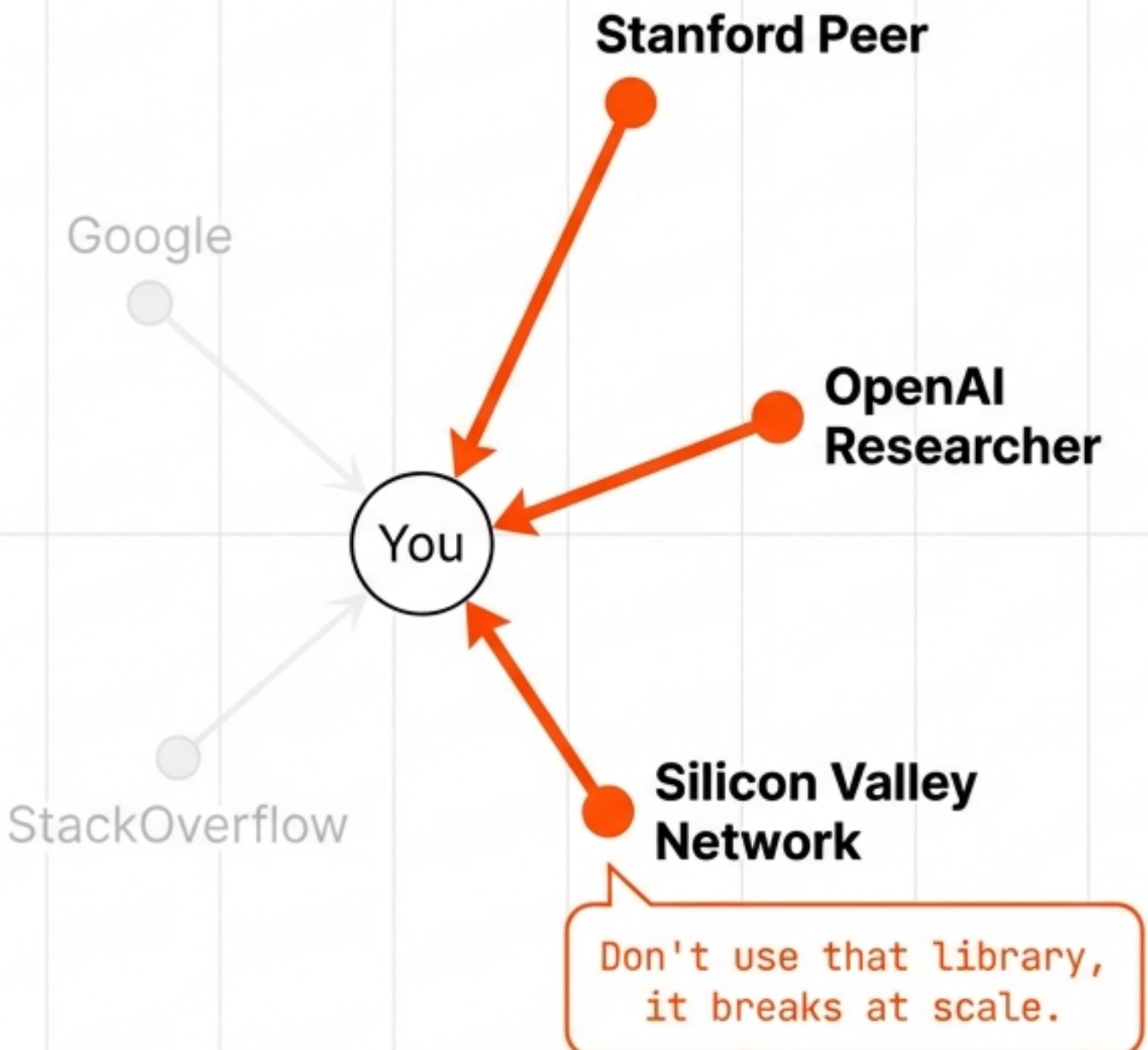
세상에서 가장 중요한 AI 정보는 구글이나 arXiv에 없습니다.
실리콘밸리의 새벽 2시 텔레그램 메시지 속에 있습니다.

Human Network:

논문이 인쇄되기 전에 작동 여부를 묻는 짧은 통화가
프로젝트의 승패를 가릅니다.

Andrew Ng's Advice:

어떤 회사 브랜드(이름표)를 다느냐보다, 누구와 함께 일하며
어떤 '연결 조직' 속에 있느냐가 당신의 성장 속도를 결정합니다.

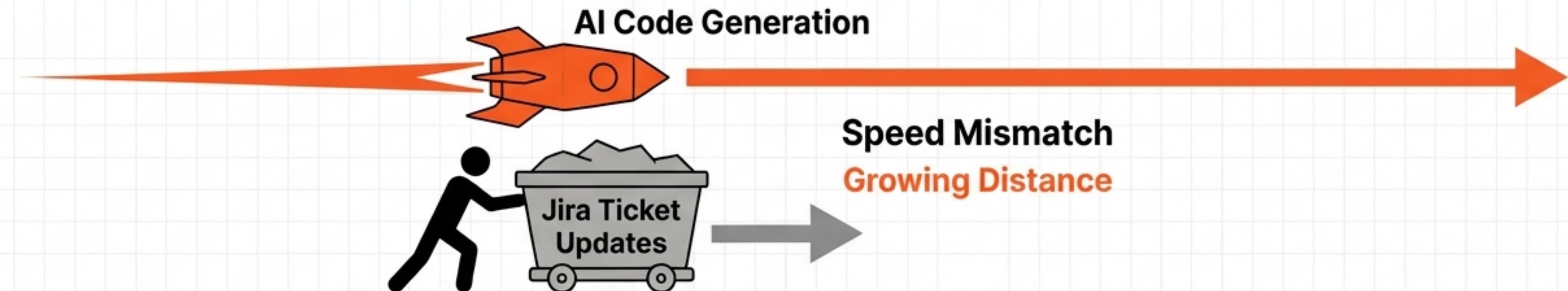


Jira는 우리를 실패하게 만듭니다: 입력 노동의 종말.

속도의 불일치: AI로 코드는 광속으로 짜는데, 티켓 업데이트와 상태 관리는 수작업으로 합니다.

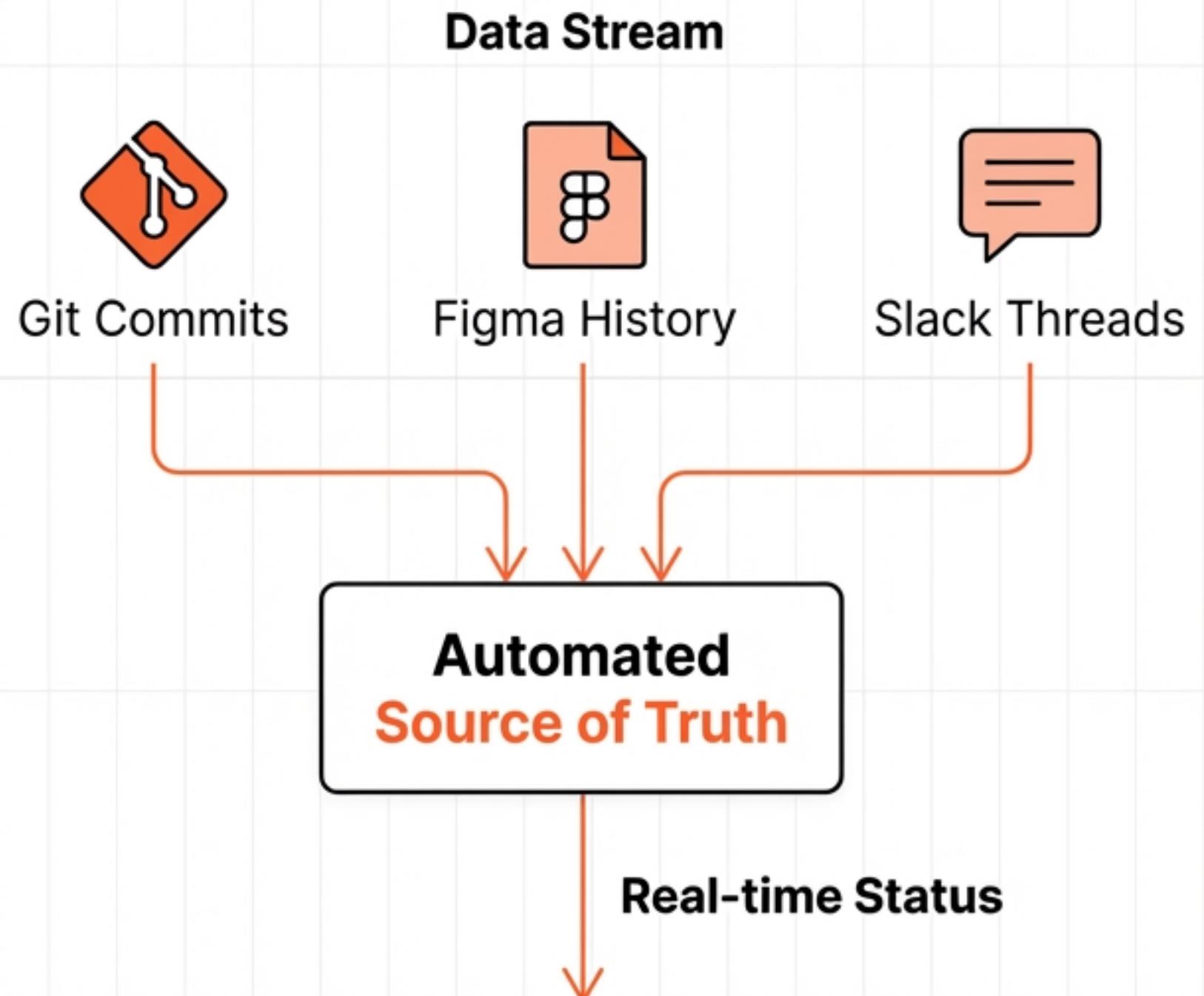
Work about Work: "일을 위한 일"이 엔지니어의 몰입(Flow)을 방해합니다. 입력을 강요할수록 데이터는 부정확해지고, 프로젝트는 느려집니다.

Yozm IT Insight: "우리는 더 나은 제품을 만들기 위해 모였는가, 아니면 티켓을 관리하기 위해 모였는가?"



입력하지 않아도 흐름이 보이는 관리: ‘실행’이 곧 기록입니다.

- **Source of Truth**: 개발자의 진척도는 Jira 티켓이 아니라 Git 커밋에 있고, 디자인은 Figma 히스토리에 있습니다.
- **Automated Tracking**: 실무자가 도구에 들어와서 ‘입력’하게 만들지 마십시오. 도구가 실무자의 흔적을 자동으로 수집해야 합니다.
- **The Future**: 관리는 ‘시키는 것’이 아니라, 이미 일어난 실행을 읽고 판단하는 것입니다.



AI 네이티브 엔지니어의 생존 체크리스트.



Tech (기술)

바로 코딩하지 마십시오. R-P-I (리서치-플래닝-구현) 프레임워크로 컨텍스트를 장악하십시오.



Mindset (태도)

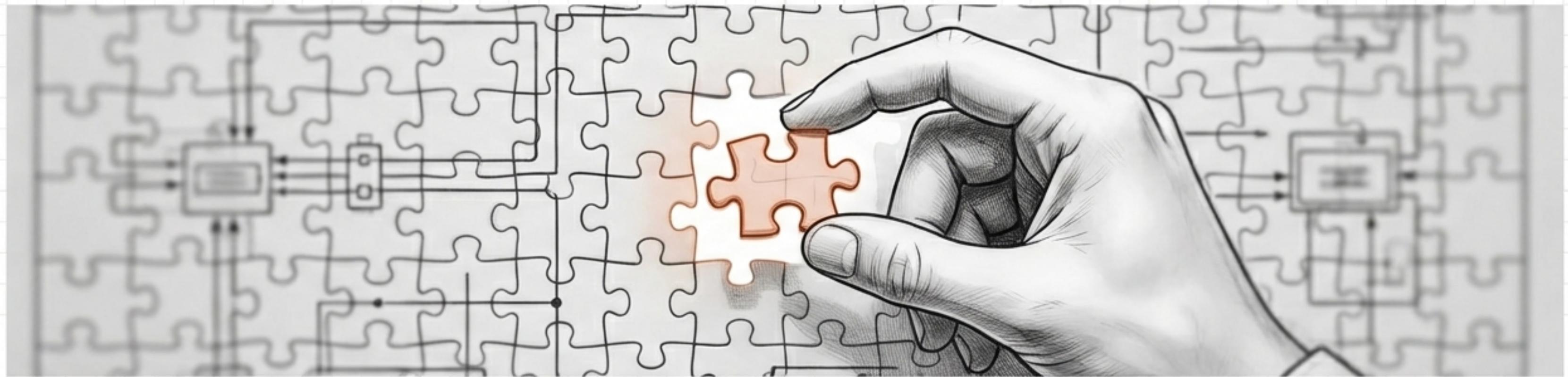
부품이 되지 마십시오. 기획과 구현을 통합하는 프로덕트 엔지니어가 되십시오.



Process (협업)

입력 노동을 거부하십시오. 실행(Git/Figma)이 곧 기록이 되는 자동화된 워크플로우를 구축하십시오.

생각은 아웃소싱할 수 없습니다. (You Cannot Outsource Thinking.)



AI가 코드의 대부분을 작성할 때, 시스템을 진정으로 이해하고,
위험한 경계를 설정하고, 무엇을 만들지 결정하는 것은 오직 당신뿐입니다.

코드는 무한하지만, 당신의 통찰은 대체 불가능합니다.