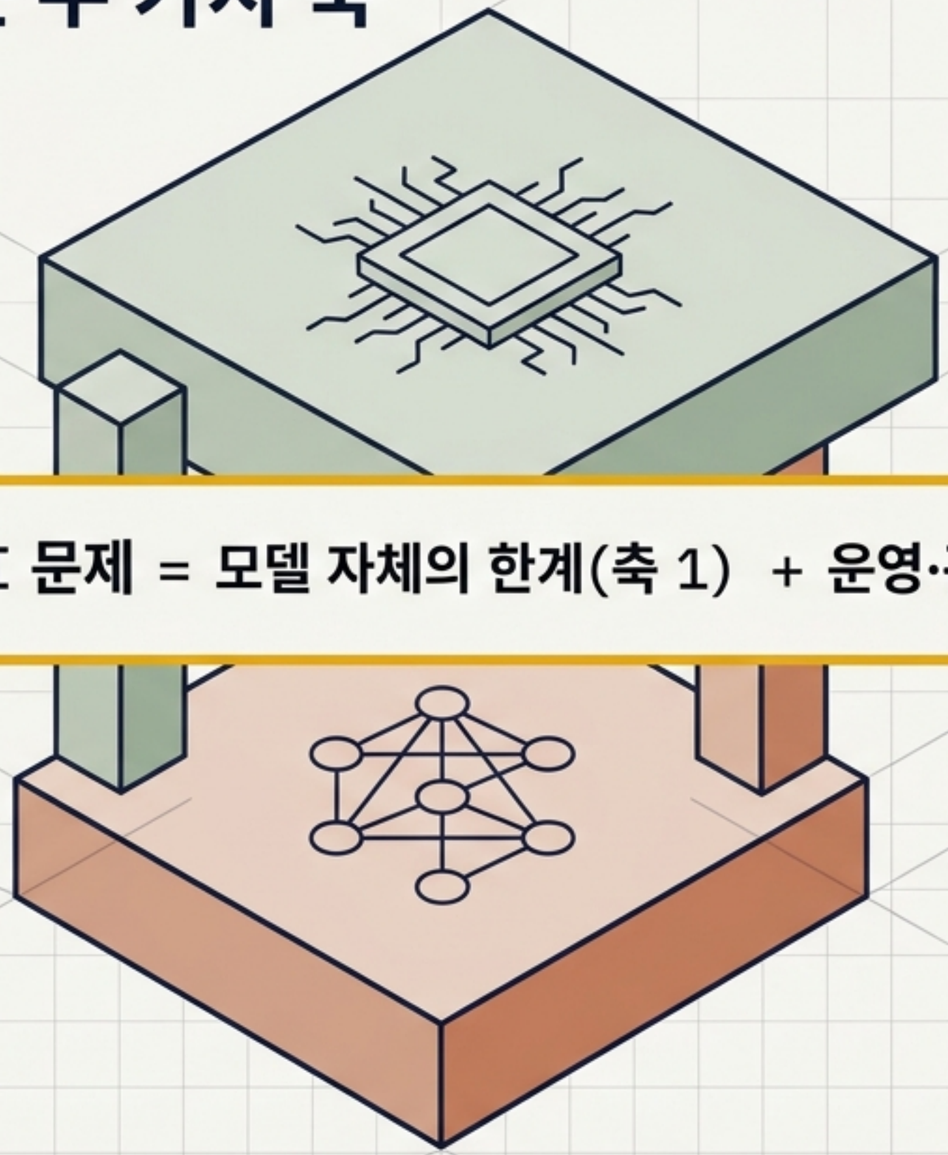
개발자가 겪는 AI 문제의 본질과 SpecKit의 해결책

[SYSTEM_BLUEPRINT: AI_WORKFLOW_RESOLUTION]

AI 도입 실패를 만드는 두 가지 축

축 1 — AI 모델 자체의 대화·추론 한계
(모든 모델의 공통적 특성)



축 2 — 팀 개발 워크플로우의 구조적 한계
(관리 방식의 부재)

개발자가 겪는 AI 문제 = 모델 자체의 한계(축 1) + 운영·관리의 한계(축 2)

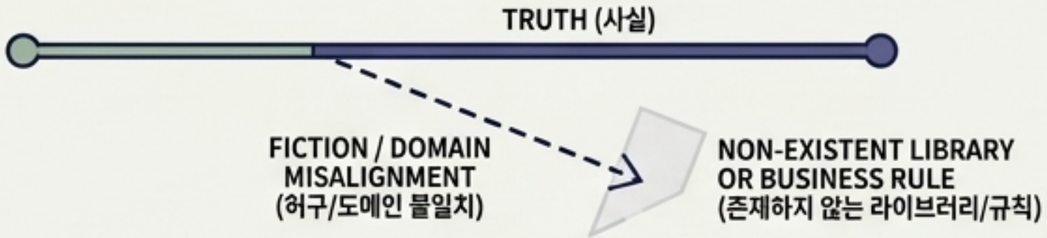
**축 1은 완화의 대상이지만,
축 2는 스펙 기반 관리로
근본적 해결이 가능합니다.**

축 1: 세션 진행 중 발생하는 모델의 본질적 한계

맥락 오염 (Context Pollution)

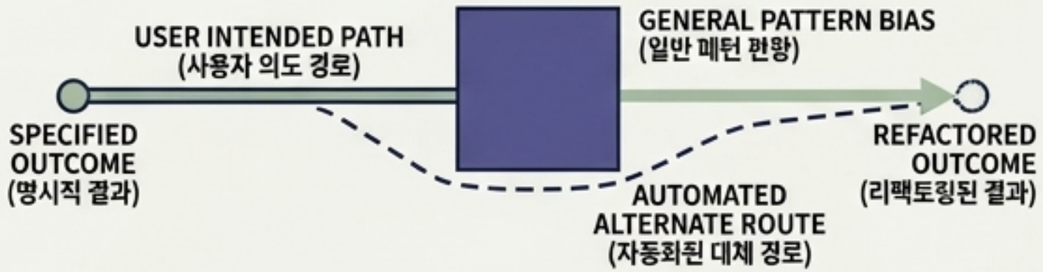
대화가 길어질수록 초반에 명확히 준 지시나 제약조건이 서서히 흐려지고 희석되는 현상.

할루시네이션 (Hallucination)



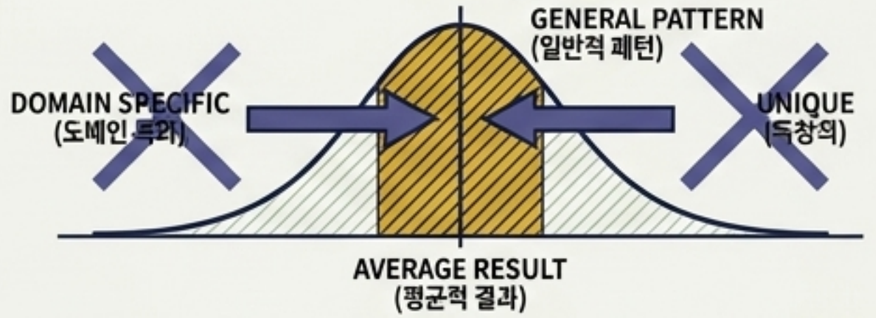
존재하지 않는 라이브러리를 지어내거나(사실적), 우리 서비스만의 특수한 비즈니스 규칙과 어긋나는 코드(도메인)를 확신에 차서 생성.

지시 무시 (Instruction Override)



명시적 지시보다 학습 데이터 내의 '일반적으로 좋은 패턴'을 우선시하여 사용자 의도와 다르게 리팩토링.

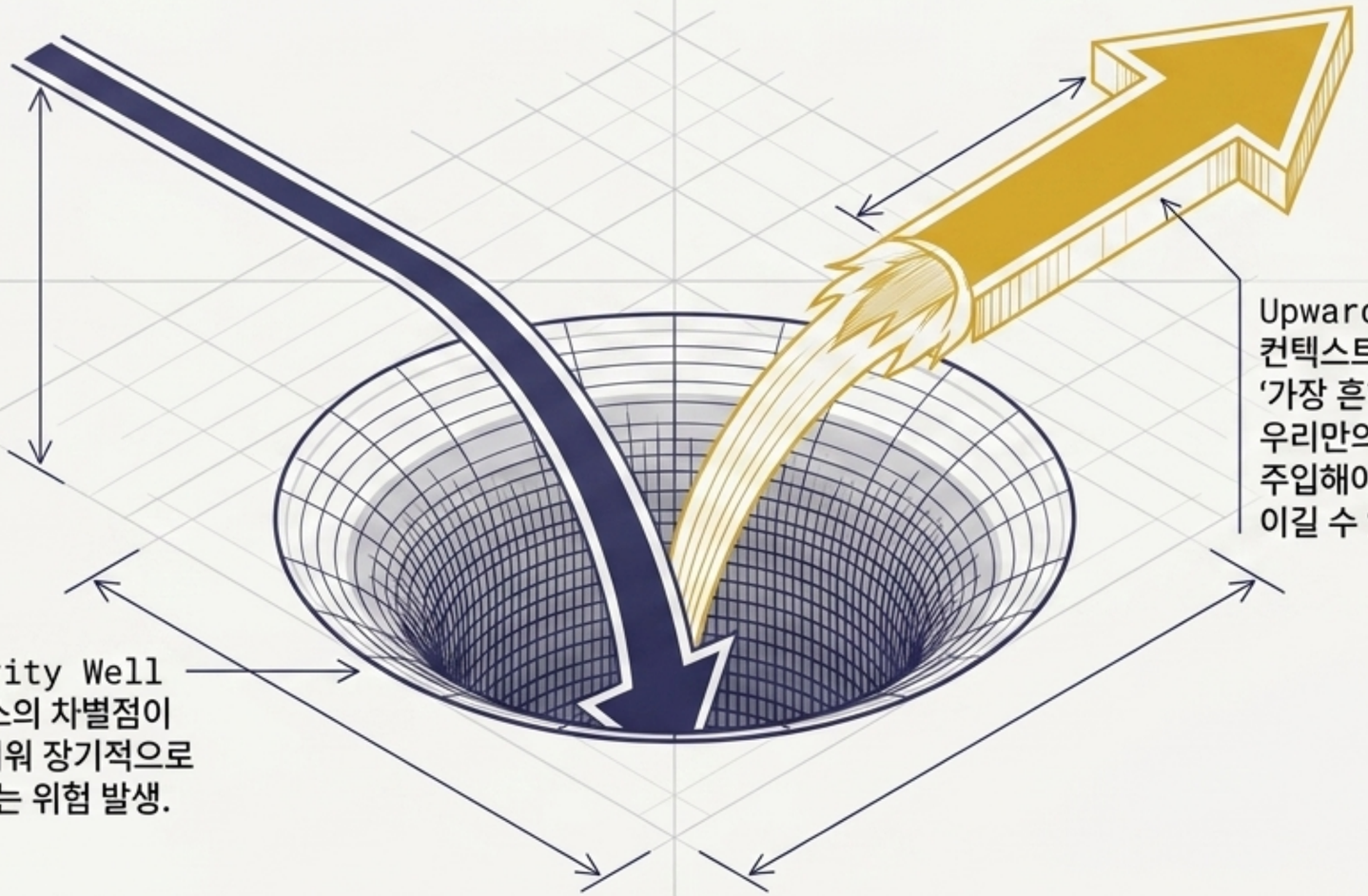
평균수렴 (Regression to the Mean)



도메인 특화 결과 대신 방대한 학습 데이터의 일반적 패턴으로 귀결.

가장 발견하기 어려운 치명적 오류: 평균수렴

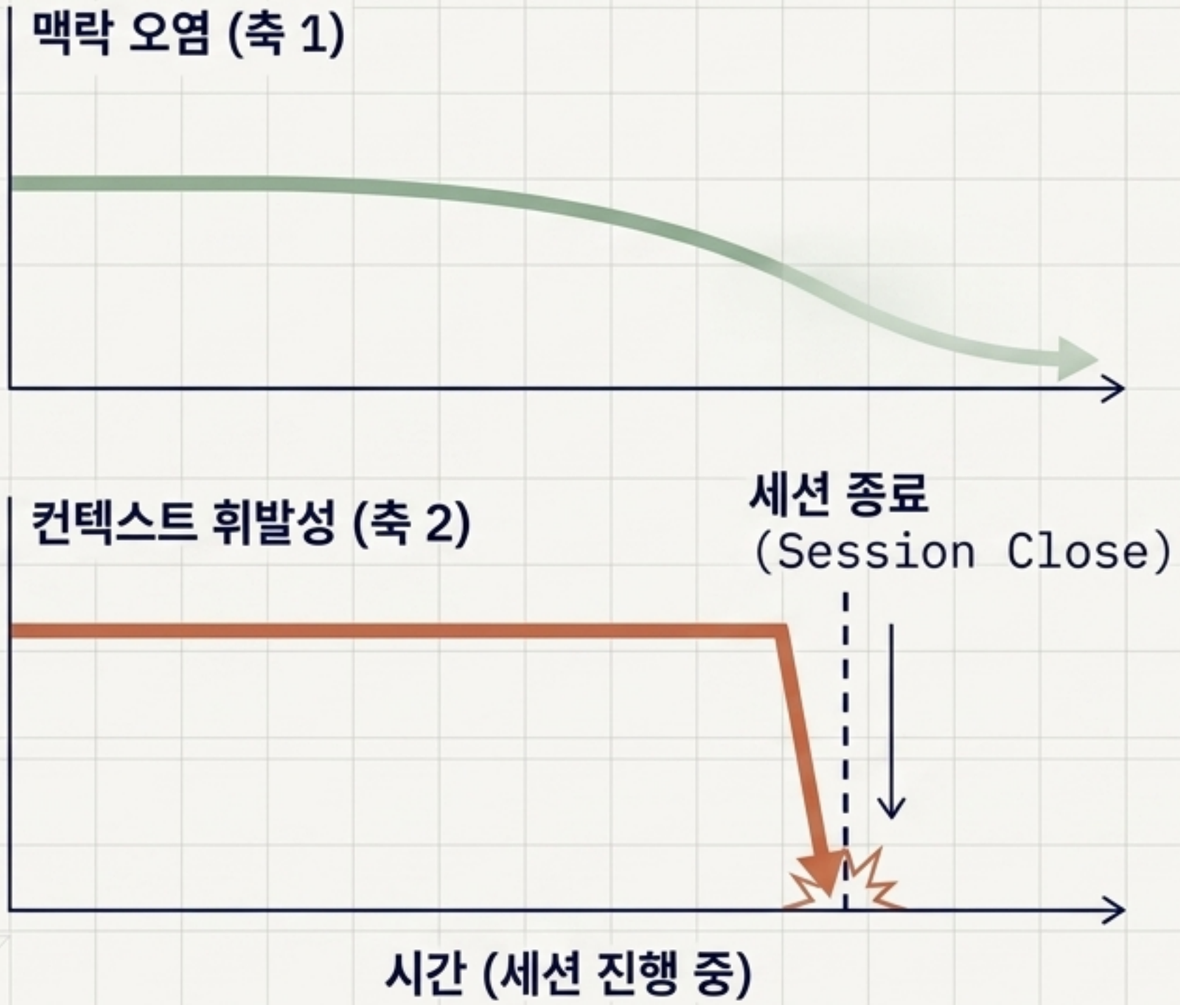
AI 출력 ≈ 학습 데이터의 평균 패턴 [Gravity] + 제공된 컨텍스트 보정값 [Thrust]



Upward Thrust Arrow
컨텍스트 보정값이 작으면
'가장 흔하고 무난한 방식'으로 수렴.
우리만의 규칙을 컨텍스트로 명확히
주입해야만 평균으로 끌려가는 힘을
이길 수 있음.

Gravity Well
코드는 정상 작동하지만 서비스의 차별점이
사라짐. 검수에서 걸러지기 어려워 장기적으로
경쟁사와 동질화되는 위험 발생.

축 2: 세션 종료 후 발생하는 워크플로우의 구조적 한계



지식 사일로 (Knowledge Silo)

개인의 채팅 히스토리에만 해결책이 고립되어 팀 자산화 불가.

모델 종속성 (Model Dependency)

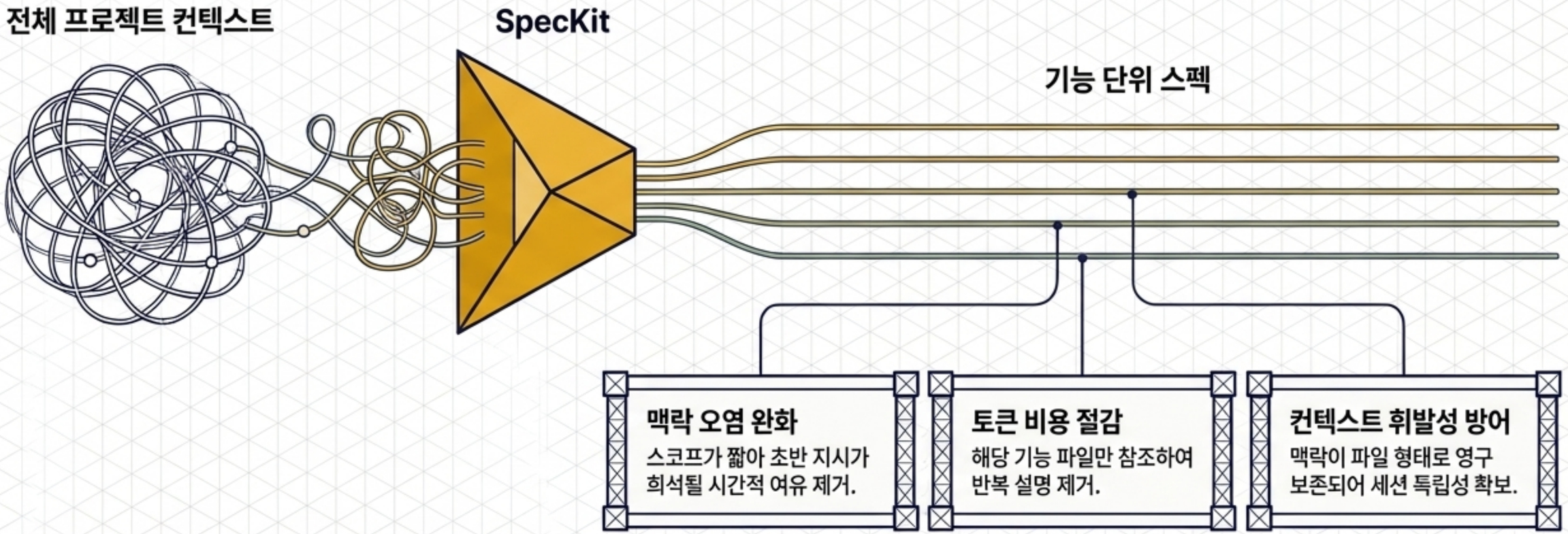
특정 모델이나 IDE에 최적화된 프롬프트로 인해 타 환경에서 이식성 저하.

토큰 비용 (Token Costs)

세션마다 동일한 배경 설명을 반복 입력하며 발생하는 불필요한 비용.

단 하나의 구조적 전환: 스펙의 기능 단위 분리

> “스펙을 기능 단위로 쪼개서 작업하니, 전체 프로젝트 컨텍스트를 매번 로딩하지 않아도 해당 기능에 필요한 최소한의 맥락만으로 정확한 결과를 얻을 수 있다.”



SpecKit Resolution Matrix: 8대 문제 통합 해결망

문제	SpecKit의 해결 메커니즘
[축 1] 맥락 오염	기능 단위로 스코프를 좁혀 대화 길이 자체를 최소화
[축 1] 할루시네이션	도메인 규칙을 스펙에 명문화하여 AI의 추측 여지를 제거
[축 1] 지시 무시	지시를 문서화된 스펙으로 고정하여 매 세션마다 동일하게 재주입
[축 1] 평균수렴	우리만의 규칙을 컨텍스트로 명확히 주입하여 일반 패턴으로의 수렴을 방지
[축 2] 컨텍스트 휘발성	스펙/계획/노트 파일이 세션 종료와 무관하게 영구 보존
[축 2] 지식 사일로	Git 기반 마크다운 저장으로 팀 전체가 공유·재사용 가능
[축 2] 모델 종속성	특정 모델/IDE에 종속되지 않는 범용 마크다운 구조
[축 2] 토큰 비용	필요한 스펙 파일만 로딩하여 반복 설명 제거

대화 중의 흔들림(모델)과 팀 단위의 흔들림(워크플로우) — SpecKit은 가장 무서운 문제인 '평균수렴'을 방어하는 완벽한 스펙 기반 체계입니다.