7회차 · 165분

하이퍼 애자일

조율의 주체를 사람에서 에이전트로 옮기면 무엇이 달라지나

발표자 노트 · 01

여는 말

- 청중이 개발조직입니다. 애자일 경험이 있는지 먼저 물어보세요.
- 이 방법론을 “애자일이 틀렸다”로 소개하면 반발합니다. 조건이 바뀐 것이라고 하세요.

먼저 인정할 것

2주 스프린트는 무엇에 맞춘 숫자였나

사람이 모이고, 정하고, 만들고, 다시 모이는 데 걸리는 시간

계획 회의	**실제 작업** 여기만 산출물	리뷰 대기	데모 · 회고

발표자 노트 · 02

2주의 정체

- 개발 자체의 속도가 아니라 사람의 협업 속도에 맞춘 숫자입니다.
- 인원을 늘려도 안 빨라집니다 — 소통 경로가 제곱으로 늘어납니다(5명 10개, 10명 45개).
- 1주로 줄이면 조율 비율이 오히려 올라갑니다.

HyperTeams

무엇이 바뀌나

조율의 주체를 옮깁니다

애자일

작업 단위
기능(feature)

사이클
1~2주

개발자 산출물
코드

병목
사람의 조율

코드의 성격
자산

하이퍼 애자일

작업 단위
✓ 이슈 하나 (마이크로 스프린트)

사이클
✓ 분 · 시간

개발자 산출물
✓ 의도

병목
✓ 의도의 명확성

코드의 성격
✓ 소모품

짧은 피드백 · 변화 수용 · 동작하는 결과물 우선은 그대로 갑니다. 사람의 판단은 더 중요해집니다.

발표자 노트 · 03

바뀌는 지점

- 병목은 사라지지 않고 이동합니다 — 애매한 요구사항을 넣으면 애매한 결과가 아주 빠르게 나옵니다.
- 각 행이 뒤의 개별 장 주제입니다.

개념 1

마이크로 스프린트 — 핵심은 크기가 아니라 독립성

쪼갰지만 의존적

1. DB 스키마 변경
2. API 수정
3. 화면 연결

→ 한 줄로 서 있음. 쪼갠 의미 없음

독립적

사진 업로드

비밀번호 재설정

탈퇴 마스킹

→ 동시에 진행. 실패 범위는 하나

발표자 노트 · 04

마이크로 스프린트

- 가로(층)로 자르지 말고 세로(기능)로 자르라는 게 핵심입니다.
- 판정 기준: 따로 완료·테스트·배포할 수 있는가. 세 번째가 가장 강합니다.
- 억지로 나누지 마세요 — 마이그레이션처럼 본질적으로 순차인 것도 있습니다.

개념 2

IntentOps — 산출물이 코드에서 의도로

- 명세 — 자연어로 쓰되 해석의 여지를 남기지 않습니다
- 테스트 — 실행 가능한 합격 기준. “완료”의 유일한 정의
- 코드는 일시적 — 최적화·스택 이동은 같은 의도로 다시 생성

의도와 테스트가 원본이고 코드는 파생물입니다. 기존에는 이 관계가 뒤집혀 있어 문서가 늘 뒤처졌습니다.

발표자 노트 · 05

IntentOps

- 솔직하게 말하세요 — “애매하게 시작하기”가 불가능해집니다. 적응이 필요합니다.
- TDD와 비슷하지만 코드를 사람이 쓰지 않고, 코드의 지위가 파생물이라는 점이 다릅니다.
- 작게 나누면 완화됩니다. 큰 기능을 한 번에 명세하려니 어려운 것입니다.

책임 1 — 명세

해석의 여지가 곧 버그입니다

이렇게 쓰면

요청
장바구니에 쿠폰을 적용할 수 있게

중복
안 적음 → AI가 알아서 정함

만료
안 적음

최소 금액
안 적음

이렇게 씁니다

요청
✓ 쿠폰 코드를 입력해 할인을 적용한다

중복
✓ 쿠폰은 하나만 적용된다

만료
✓ '사용 기한이 지난 쿠폰입니다' 표시

최소 금액
✓ '3만원 이상 주문 시 사용 가능' 표시

자연어로 쓰되 해석의 여지를 남기지 않습니다. 좋은 지시의 네 요소를 코드 작업에 적용한 것입니다.

발표자 노트 · 06

명세

- 왼쪽을 넣으면 AI 가 “중복 불가”를 스스로 정하고, 그 결정이 기획 의도와 다를 수 있습니다.
- 이 슬라이드에서 참석자에게 자기 팀의 최근 티켓 하나를 오른쪽 형태로 고쳐 쓰게 하면 실습이 빨라집니다.
- 분량이 늘어나는 것을 부담스러워합니다 — 뒤에서 되돌아오는 왕복이 사라진다는 점을 함께 말하세요.

책임 2 — 테스트

테스트가 "완료"의 유일한 정의입니다

- 유효한 쿠폰 적용 → 할인가 표시
- 두 번째 쿠폰 적용 시도 → 거부 + 안내 문구
- 만료 쿠폰 → '사용 기한이 지난 쿠폰입니다'
- 2만원 주문에 3만원 쿠폰 → '3만원 이상 주문 시 사용 가능'
- 적용 후 총액 = 원가 – 할인액

다섯 개가 통과하면 끝난 것이고, 통과 안 하면 안 끝난 것입니다. "거의 다 됐다"가 없습니다.

발표자 노트 · 07

테스트

- 명세를 판정 가능한 형태로 옮긴 것뿐입니다. 새로 만드는 문서가 아닙니다.
- 이 다섯 줄이 프로세스 3 의 판정자가 됩니다 — 뒤의 자가 치유 루프가 이걸 돌립니다.
- “80% 됐어요”가 사라지는 것이 관리 측면에서 가장 큰 변화입니다. 진척 보고의 성격이 바뀝니다.

솔직하게

더 어려워지는 것도 있습니다

기존

애매하게 시작
가능 — 만들면서 정함

만들면서 이해
흔한 방식

부분 완료
“80% 됐어요”

IntentOps

애매하게 시작
✓ 불가능

만들면서 이해
✓ 앞에서 이해해야 함

부분 완료
✓ 통과 아니면 미통과

“일단 만들어보면서 요구사항을 이해하는” 방식이 안 통합니다. 이 회차에서 가장 저항이 큰 지점입니다.

발표자 노트 · 08

무엇이 어려워지나

- 이 슬라이드를 빼지 마세요. 좋은 점만 말하면 현장에서 신뢰를 잃습니다.
- 탐색이 필요한 일에는 여전히 프로토타입이 맞습니다 — 방법론을 전면 적용하라는 말이 아닙니다.
- 여기서 나오는 반발이 다음 실습의 재료입니다. 어느 백로그가 "애매해서 못 쪼갠다"인지 적어두게 하세요.

개념 3

유동적 소프트웨어 — 레거시의 정의가 바뀝니다

기존

레거시
손대기 무서운 오래된 코드

자산
코드

리뷰 대상
코드 변경

기술 부채
나쁜 코드

유동적 소프트웨어

레거시
✓ 의도로 다시 만들면 되는 것

자산
✓ 명세 + 테스트

리뷰 대상
✓ 의도 변경

기술 부채
✓ 테스트가 얇은 영역

테스트가 얇으면 재생성해도 맞는지 확인할 수 없어, 그 영역만 예전처럼 굳습니다.

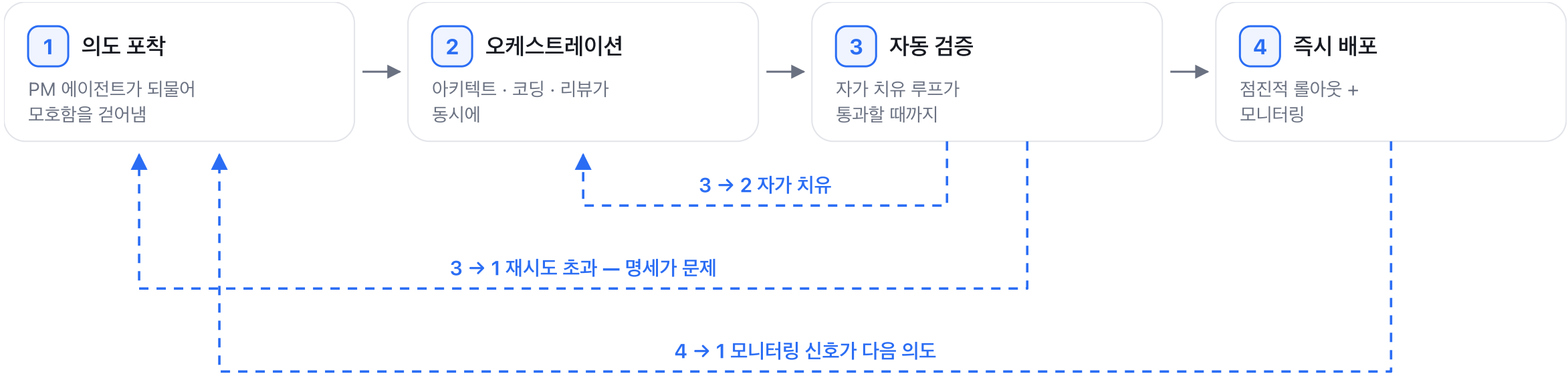
발표자 노트 · 09

유동적 소프트웨어

- 현실적으로 전부 재생성하는 조직은 드뭅니다. 새로 만드는 것부터 이 방식으로 하세요.
- 조심할 것 — 테스트에 없는 암묵적 계약(응답 필드·정렬 순서·오류 문구)이 재생성 후 깨집니다.

프로세스

4단계



실선 셋은 순서일 뿐입니다. 되돌아오는 점선 셋이 이 그림의 내용입니다 — 다음 장에서 하나씩 봅니다.

발표자 노트 · 10

4단계

- 1단계에서 남긴 애매함은 저절로 해결되지 않습니다 — 테스트까지 그 기준으로 만들어져 안 걸립니다.
- “이미 할인 중인 상품에도 적용되나요?” 같은 질문이 이 단계의 가치입니다.

프로세스

점선 셋이 이 그림의 내용입니다

01

3 → 2 자가 치유

테스트가 실패하면 사람을 부르지 않고 코드를 다시 만듭니다. 루프의 기본 동작

02

3 → 1 되돌아감

몇 번 고쳐도 안 되면 코드가 아니라 명세나 테스트가 틀린 것입니다. 여기서 사람이 들어옵니다

03

4 → 1 루프가 닫힘

배포 후 모니터링 신호가 다음 사이클의 의도가 됩니다

한 바퀴가 분·시간 단위입니다. 2주가 아닙니다.

발표자 노트 · 11

세 개의 점선

- 실선 넷보다 점선 셋이 이 방법론의 내용입니다. 앞의 화살표만 보면 그냥 폭포수처럼 보입니다.
- 두 번째 점선이 사람의 자리입니다 — 재시도 초과가 "명세를 다시 보라"는 신호입니다.
- 세 번째 점선이 없으면 그냥 빠른 개발입니다. 운영 신호가 다음 의도가 되어야 루프가 닫힙니다.

익숙한 4단계

분석 · 설계 · 구현 · 시험은 어디로 갔나

익숙한 단계

분석
요구사항 회의

설계
구현 전 단계

구현
사람이 코드를

시험
맨 뒤 관문

배포
단계 밖 이벤트

어디로 갔나 · 무엇이 달라졌나

분석
✓ 프로세스 1 — 회의록이 아니라 문답이 그대로 명세

설계
✓ 프로세스 2 안의 아키텍트 — 구현·리뷰와 동시에

구현
✓ 프로세스 2 — 코드는 자산이 아니라 파생물

시험
✓ 앞으로 당겨짐 — 명세와 함께 정하고 뒤에서는 판정만

배포
✓ 프로세스 4 — 루프 안

사라진 게 아니라 자리가 바뀐 것입니다. 순서가 아니라 조율이 바뀌었습니다.

발표자 노트 · 12

4단계는 어디로

- 폭포수로 일해온 조직이 가장 먼저 묻는 질문입니다. 이 표 하나로 "없앤 게 아니다"가 정리됩니다.
- 설계가 사라졌다는 오해를 반드시 푸세요 — 아키텍트 역할이 그대로 있고, 다만 구현이 끝나기를 기다리지 않습니다.
- 사람이 설계에 개입하는 지점은 둘로 정해져 있습니다: 설계 방향이 명세와 어긋날 때, 되돌릴 수 없는 결정(스키마·외부 계약)일 때.
- 시험이 앞으로 당겨진 이유는 하나입니다 — 사람이 완료를 판정하지 않으려면 판정 기준이 시작 시점에 있어야 합니다.

프로세스 1

의도 포착 — 왜 사람이 아니라 에이전트가 묻나

사람 PM

시점
회의를 잡아야 함

빠뜨림
익숙한 영역은 건너뜀

기록
회의록에 흩어짐

PM 에이전트

시점
✓ 즉시

빠뜨림
✓ 체계적으로 훑음

기록
✓ 문답이 그대로 명세가 됨

산출물은 해석의 여지가 없는 명세입니다 — “범위 밖”이 적혀 있는 것이 핵심입니다.

의도 포착

- 요청자가 “생각 안 해봤다”고 답하면 좋은 신호입니다. 되묻기의 목적이 정확히 그것을 찾는 것입니다.
- 여기서 남긴 애매함은 뒤에서 AI가 하나를 골라 진행하고, 테스트도 그 기준으로 만들어져 검증에서 안 걸립니다.
- 되묻는 질문을 팀의 목록으로 가지세요 — 경계값 · 예외 · 권한 · 실패 시 동작.

프로세스 2

오케스트레이션 — 역할을 나누는 이유 셋

01

컨텍스트가 깨끗해짐

각자 자기 일에 필요한 것만 봅니다

02

관점이 분리됨

만든 사람이 자기 코드를 검토하지 않습니다

03

병렬이 가능해짐

같은 작업 공간을 보므로 전달 손실이 없습니다

사람 팀의 위계를 옮기면 순차 구조가 되어 대기가 생깁니다. 역할을 계속 늘리면 조율 비용이 다시 생깁니다.

오케스트레이션

- 아키텍트 · 코딩 · 리뷰 셋이면 대개 충분합니다. 없애려던 문제를 에이전트 구성에서 다시 만들지 마세요.
- 리뷰 에이전트에게 무엇을 볼지 알려주세요 — “보안·품질”만으로는 매번 다른 것을 봅니다.
- 사람은 의도를 주는 앞과 판정하는 뒤에 있습니다. 가운데에 서면 병목이 됩니다.

가장 취약한 지점

얕은 테스트는 사람보다 AI에게 더 위험합니다

테스트

쿠폰 적용 후 총액이 원가보다 작아야 한다

루프가 만든 통과 코드

```
총액 = 원가 - 1원
```

사람은 “이건 아니지”라고 알지만 루프는 모릅니다. 루프의 목표는 **테스트 통과**이지 의도 충족이 아닙니다.

발표자 노트 · 15

자가 치유 루프의 함정

- 이 장이 이 회차에서 가장 중요합니다. 반드시 시간을 쓰세요.
- 대처 — 테스트를 명세로 취급 · 단위/통합/E2E 겹치기 · 샌드박스 · 재시도 횟수 제한.
- 테스트를 코드에 맞춰 고치는 순간 방어선이 사라집니다.

배포

작고 자주가 오히려 안전합니다

큰 배포 · 드물게

변경량
한 번에 100개

원인 추적
어려움

롤백 범위
100개가 다 사라짐

작은 배포 · 자주

변경량
✓ 1개

원인 추적
✓ 자명함

롤백 범위
✓ 1개

전제가 많은 단계입니다 — 자동 배포·빠른 롤백·실시간 모니터링·점진적 롤아웃. 없으면 1~3만 도입하세요.

발표자 노트 · 16

즉시 배포

- 카나리·블루그린 — 어느 쪽이든 되돌리는 경로가 미리 있습니다.
- 모니터링 신호가 다음 사이클의 의도가 되며 루프가 닫힙니다.

가드레일

속도를 감당하게 하는 조건

01

Human-in-the-Loop

되돌릴 수 있는가로 승인 지점 결정. 너무 많으면 애자일로 회귀

02

환각 통제

리뷰가 아니라 결정론적 테스트 스위트. 층을 겹칠 것

03

비용 관리

모델 라우팅 — 난이도에 모델을 맞춤

가드레일은 브레이크가 아닙니다. 그 속도를 감당할 수 있게 만드는 조건입니다.

발표자 노트 · 17

가드레일

- 승인이 걸리는 곳 — 돈이 움직이는 경로 · 권한 · 외부로 나가는 데이터 · 되돌리기 어려운 변경.
- 사이클이 짧을수록 모델 선택이 자주 일어나 효과가 누적됩니다.

도입 전

자가 점검 — 하나라도 비면 그것부터

- 테스트 자동화가 신뢰할 만한가 — 절대 조건
- 배포 파이프라인이 자동인가
- 롤백이 빠른가 (분 단위)
- 모니터링이 실시간인가
- 승인 지점이 합의됐는가

테스트 문화가 없는 조직에는 권하지 않습니다 — 속도만 올라가고 품질 판정은 아무도 못 하게 됩니다.

발표자 노트 · 18

도입 전 점검

- 첫 줄만 절대 조건이고 나머지는 부분 도입이 가능합니다.
- 방법론 전체를 도입할 필요 없습니다. 개별 요소(작게 쪼개기·테스트를 완료 기준으로)부터 쓸 수 있습니다.

실습 · 40분

우리 백로그를 쪼개보기

1
한 문장으로
제목이 아니라 문장.
“그리고”가 있으면 이미
두 개

2
접속사로 자르기
네 줄 이상 나와야 합니다

3
화살표 그리기
“이게 끝나야 저걸 시작”.
대개 여기서 한 줄로 서
있음을 봅니다

4
세로로 다시
층이 아니라 기능으로.
남으면 계약을 먼저 고정

5
배포로 판정
이것만 따로 내보내도 안 깨지나

“나가긴 하는데 화면이 반쪽”은 스위치를 두고 꺼둔 채 내보내면 됩니다 — 이 제약이 의존성의 절반을 만듭니다.

실습 진행

- 끝난 일 말고 **진행 중인** 일감을 고르게 하세요. 끝난 일은 결과를 알아 깔끔하게 나뉩니다.
- 3단계에서 화살표가 한 줄로 이어지는 것이 정상 결과입니다. 실패가 아니라 발견이라고 말해주세요.
- 화살표를 하나도 못 없앴다면 진짜 순차이거나 아직 층으로 자르고 있는 것입니다.
- 실습용 일감이 없으면 “회원 탈퇴 기능 개선”을 쓰게 하세요.

실습 · 3~4단계

화살표를 없애는 법

층으로 자른 것

3. 사유를 저장할 자리

↓ 이게 끝나야 시작

1. 탈퇴 화면에서 고르기

↓ 이게 끝나야 시작

4. 관리자 통계 화면

→ 다섯으로 쪼갰는데 결국 한 줄

기능으로 다시 자른 것

A. 탈퇴 사유 받기

저장 자리 + 화면 + 저장까지 한 덩어리

B. 탈퇴 통계 보기

A 가 만든 자리를 읽어 화면에 표시

→ 저장 형태만 5분 먼저 고정하면 A · B 동시에

같은 일감입니다. 자르는 방향만 바꿨는데 화살표가 사라졌습니다 — 이 5분이 며칠을 벌어줍니다.

발표자 노트 · 20

층 vs 기능

- 왼쪽이 참석자 대부분이 그리게 되는 그림입니다. 실패가 아니라 발견이라고 말해주세요.
- 오른쪽으로 옮기는 요령은 하나입니다 — 한 조각이 저장부터 화면까지 혼자 끝까지 가게 하는 것.
- A · B 사이에 남는 화살표는 "계약(저장 형태)만 먼저 고정"으로 끊습니다. 전부를 합의하는 게 아닙니다.

실습 · 40분

검증 관문을 설계해보기

1
지금 누가 무엇을
검수자가 눈으로 보는 것을
이유까지 적습니다

2
세 갈래로
기계가 판정 · 기계가 후보만 · 사람만

3
관문 셋
초 단위 · 분 단위 · 사람. 실패하면
무엇이 일어나나

4
되돌리기
되돌릴 수 없으면 사람 관문은 필수

"사람만"이 절반을 넘으면 검증이 어려운 게 아니라 아직 안 쪼갠 것입니다. "문서가 맞나"는 사람, 그 안의 "금액이 표와 맞나"는 기계입니다.

발표자 노트 · 21

실습 진행

- 1단계에서 “품질을 본다”가 나오면 무엇을 보면 나쁜 줄 아는지 되물으세요.
- 실패했을 때 아무 일도 안 일어나면 관문이 아니라 로그입니다 — 이 말을 꼭 하세요.
- 되돌리기 비용이 관문의 두께를 정합니다. 분 단위면 느슨하게 하고 빨리 내보내는 편이 낫습니다.
- “왜 보나”가 빈 항목은 예전 사고의 흔적이거나 아무도 이유를 모르는 관성입니다.

실습 · 3단계

관문 셋은 이렇게 좁아집니다

관문 1 · 기계 · 초

형식 · 필수값 · 합계 · 링크 유효성

실패하면 **아예 진행 안 됨**

↓ 여기까지 통과한 것만

관문 2 · 기계 · 분

평소와 다른 값 · 금칙어 · 말투

실패하면 **사람에게 표시**

↓ 여기까지 통과한 것만

관문 3 · 사람 · 필요시

이 시점에 이 말을 해도 되는가

실패하면 **되돌리거나 보류**

"실패하면" 칸이 비면 관문이 아니라 로그입니다. 그리고 관문의 두께는 되돌리기 비용이 정합니다 — 되돌릴 수 없으면 관문 3은 필수입니다.

발표자 노트 · 22

관문 셋

- 좁아지는 폭이 요점입니다 — 사람이 전수 검사를 하지 않고, 앞의 기계 둘이 걸러낸 것만 봅니다.
- 공항 보안검색과 같은 구조라고 하면 대부분 바로 이해합니다.
- 관문 1 에 무엇을 넣을지 못 정하겠다면 1단계 표의 “왜 보나”가 비어 있는 항목부터 보세요.

정리

오늘 가져가실 것

1. 2주는 사람의 협업 속도에 맞춘 숫자였습니다
2. 쪼개는 것보다 의존을 끊는 것이 핵심
3. 테스트가 완료의 유일한 정의 — 얕으면 루프가 통과를 만들어냅니다
4. 갖춰진 만큼만 도입하세요

발표자 노트 · 23

정리

- AI Transformation 트랙이 끝났습니다. 이후는 제품 사용 트랙입니다.
- 일곱 회차를 관통한 한 줄을 다시 말해주세요 — 병목은 사라지지 않고 이동합니다.
- 여기서 각자 “우리 조직에서 다음 주에 할 것 하나”를 적게 하고 닫으면 전환율이 눈에 띄게 올라갑니다.