추정 못하는 사람들

시간과 비용을 맞추는 추정 규율

ThakiCloud (Hyojung Han)

추정 못하는 사람들

시간과 비용을 맞추는 추정 규율

ThakiCloud (Hyojung Han)

Contents

1 왜 추정은 항상 낮게 잡히는가

인디 개발 스튜디오를 운영하는 친구가 한때 이런 질문을 던졌다. "왜 나는 늘 부족하게 추정을 하나? 초과할 거 알면서." 이상했던 건 그 다음 문장. 일부러 낮게 부른 게 아니었다. 진심으로 믿었다. 계획을 세울 때 그 숫자를 진심으로 믿는다고 했다. 범위를 세 번이나 확인한 뒤에 3주를 말했고 그 3주는 그의 눈에는 정직한 숫자였다.

그 믿음이 이 책의 출발점이다. 추정 오차의 원인은 불성실이나 실력 부족이 아니다. 숫자를 만들어내는 과정. 과정은 바꿀 수 있다. 이 책은 1인 개발자와 스타트업 founders가 바로 쓸 수 있는 그 과정을 설치하는 안내서다. 읽고 바로 써 보고 다음 견적에서 차이를 확인하는 식이다.

1.1 인사이드 뷰

계획을 세울 때 우리는 눈앞의 프로젝트만 본다. 어떤 기능이 들어가고 어떤 기술로 하고 작업이 어떤 순서로 흐를까. 장애물이 될 만한 것은 뭐가 있겠지, 상대는 어떤 반응을 보일까. 머리 속에서 가장 매끄러운 경로를 한 번 돌리고, 거기서 나온 숫자가 견적이다.

이 방식이 인사이드 뷰다. 이번 프로젝트의 구체적 디테일에 집중하는 시선. 옳지 않다기보다, 치명적인 맹점. 인사이드 뷰에는 생각한 장애물만 들어 있다. 생각하지 못한 장애물은 빠져 있다. 그리고 프로젝트가 초과한 이유의 대부분은 후자에 속한다.

예를 하나 들겠다. 식당 앱에 모바일 주문 기능을 만든다고 하자. 디자인 1주, 개발 2주, 테스트 1주. 총 4주. 전혀 정직한 계획이다. 그런데 실제로는 결제 대행사 승인 기다리느라 3주가 걸리고 식당 대표가 메뉴 구조를 두 번 바꾸고, 기존 예약 시스템의 버그가 드러난다. 실제 9주. [추정] 이 중

어느 것도 4주 계획에 없었다. 게으름 때문이 아니라, 추정 시점에는 몰랐기 때문이다.

두 번째 예는 스타트업이다. founders가 MVP를 6주에 계획한다. 프로토타입 2주, 백엔드 2주, 통합과 테스트 2주. 그런데 유저 인터뷰가 3주를 잡아먹는다. 인터뷰이를 잡는 데 시간이 걸리니까. 범위는 "예약"에서 "예약에 좌석 선택과 리뷰까지"로 자란다. 핵심 개발자 한 명이 다른 팀으로 옮긴다. 4개월 뒤에 출시한다. [추정] 6주 계획은 거짓이 아니었다. 다만 그 계획에는 "일"만 들어 있었다. 일이 일어나는 환경은 빠져 있었다.

두 예의 공통점을 짚어 보자. 계획된 시간은 "내가 할 일"만 세고 있다. 승인 기다리기, 상대가 마음을 바꾸기, 사람이 이동하기. 이 셋은 계획에 없던 것이 아니라, 계획이라는 형식에 들어가지 않는 것들이다. 일이라고 생각하는 건 간트 차트에 칸을 만들 수 있는 것뿐이니까. 그런데 현실은 간트 차트가 아니다. 계획에 "일"만 넣으면, 초과가 필연. 초과는 예외가 아니라, 기본값이다.

1.2 플래닝 폴러시

이 현상에 대한 유명한 실험이 있다. 연구자가 졸업 논문을 끝내는 데 걸릴 날 수를 학생들에게 추정하게 했다. 결과는 늘 일관됐다. 추정이 실제보다 짧았다. 더 흥미로운 건 후속 실험이다. "지난 학기는 평균적으로 며칠 걸렸다"는 통계를 미리 알려준 집단도 여전히 부족하게 추정했다. 이런 프로젝트는 초과한다는 밖의 숫자를 알고 있었으면서도, "이번엔 다르다"고 믿은 것이다.

이것이 플래닝 폴러시, 즉 계획할 때 발생하는 낙관 편차다. 과거에 부족하게 추정했던 경험이 풍부할수록, 다음 계획은 더 단단해지는데 추정만은 낮게 나온다는 현상. 핵심은 "안다"는 게 막아주지 않는다는 점. 나는 늘 초과한다는 걸 알면서 다음 견적은 다시 낮게 나온다. 알고

있다는 느낌과 숫자를 만드는 작업은 별개의 과정이다. "초과하겠지"는 감각이고 "견적 3주"는 계산이다. 계산하는 순간 우리가 보는 것은 인사이드 뷰뿐이다.

그렇다면 "이번엔 다르다"는 믿음은 어디서 나오는가. 기억은 선별적으로 작동한다. 기한을 딱 지켜 끝낸 프로젝트는 또렷하게 남고 3주가 9주가 된 프로젝트는 "상대가 난폭했기 때문"으로 정리된다. 게다가 프로젝트마다 디테일이 다르다. 기술이 다르고 상대가 다르고 범위가 다르다. 그래서 뇌는 과거의 통계가 이번에는 적용되지 않는다고 결론낸다. 두 메커니즘, 선별 기억과 독창성 착시가 같은 결과를 만든다. 밖의 숫자는 존재하지만, 계산의 순간에는 호출되지 않는다.

한 가지 더 짚어야 할 디테일이 있다. 계획에 대한 확신이 강할수록 추정 숫자는 낮아진다. 고객에게 계획을 설명하는 순간 우리의 뇌는 이미 그것을 "판매"하고 있다. 판매는 잘 될 거라고 믿는 것이고 믿는 것은 경로를 짧게 만드는 것이다. 회의와 추정은 반대로 움직인다. 그래서 가장 자신 있는 견적일수록 정확하지 않은 경우가 많다.

여기에 한 가지를 덧붙이자. 경험도 이 문제를 고쳐주지 않는다. 10년 차 1인 개발자의 인사이드 뷰는 입문자의 것보다 훨씬 정확하다. 자기 코딩이 얼마나 걸리는지는 잘 안다. 그런데 경험이 성장하지 않는 부분이 정확히 프로젝트 밖이다. 외부 서비스 승인 시간, 상대의 검토 사이클, 벤더의 출시 일정. 이 것들은 개인 실력으로 빨라지지 않는다. 데이터로만 빨라진다. 그리고 대부분의 사람은 그 데이터를 모으지 않는다. 그래서 우리는 더 숙련될수록, 내 작업의 추정은 정확해지는데 초과를 만드는 부분의 추정은 제자리를 밟는다.

프리랜서 개발자의 얘기를 하나 더. 500만 원짜리 사이트 견적을 3주로 내고 실제로는 6주에 끝냈다. 시간당 단가가 반으로 줄어든 셈이다. 더 큰 손실은 숫자가 아니다. 상대가 이 개발자에게 부여한 첫인상은 "실력이 좋다"가 아니라 "시간을 못 잡는다"였다. 기술과 시간 관리는 별개로

평가된다. 그래서 6주가 된 그 500만 원은, 다음 500만 원의 시세도 낮췄다. 그리고 후자의 평판이 다음 견적의 가격을 결정한다.

1.3 왜 낮은 추정이 매력적인가

사회적 메커니즘도 있다. 낮은 추정치가 일을 따낸다. 같은 프로젝트에 두 개발자가 응찰하면 3주를 말한 쪽이 5주를 말한 쪽보다 유리하다. 3주에 확신이 없는데도 3주를 말한다. 그리고 습관이 형성된다. "그냥 해보면서 맞추자."

또 다른 이유는 손실의 비대칭이다. 부족하게 추정한 대가는 크다. 프로젝트가 초과하고 손님이랑 관계가 깨지고 결국 추가 작업을 공짜로 한다. 그런데 높게 추정한 대가는 작아 보인다. 고객은 "더 일찍 끝내겠지"라고만 한다. 그래서 우리는 즉각적 비용이 작은 숫자를 고른다. 낮은 숫자를.

흥미로운 건 이 비대칭이 절반은 환상이라는 점이다. 늘 높은 견적은 한 건에서는 불리해 보이지만 열 건에서는 신뢰를 만든다. 반대로 늘 초과하는 낮은 견적은 다음 일 자체를 죽인다. 입소명에 의존하는 1인 개발자나 소규모 founders라면 이걸 잘 안다. 견적을 내는 그 순간에는, 그 한 건만 보인다.

그 위에 앵커링이 한 겹 더 얹힌다. 먼저 말한 숫자가 기준점이 된다. 당신이 4주를 말하면, 상대는 그 4주를 바탕으로 일정을 세운다. 마케팅은 4주 뒤에, 팀 계획도 4주 뒤에, 예산도 4주 뒤에 배치된다. 그래서 나중에 숫자를 바꾸는 비용은, 처음에 낮은 숫자를 말한 비용보다 커진다. 우리는 일찍 낮은 숫자를 잠그고 늦게 그 숫자를 지키기 위해 애쓴다. 아이러니하게도, 3장에서 다룰 재추정 규율이 이 문제를 푸는 열쇠다. 다만 초기 견적 시점에서는 아직 그 규율을 쓰지 않고 있을 뿐이다.

추정은 당신이 하는 숫자이기도 하지만 상대의 계획이기도 하다. 상대는

당신의 숫자 주변에 출시일을 정하고 예산을 정하고 팀 일정을 정한다. 숫자가 입에서 나온 순간, 상대의 운영 사실이 된다. 그래서 낮은 추정은 당신 일정이 깨지는 문제를 넘어서, 상대의 전체 계획이 깨지는 문제다. 그 대가는 견적 금액보다 항상 크다. 초과한 주수를 돈으로 환산해도 모자란다.

1.4 4주에서 9주로 간 한 프로젝트

초과가 어떻게 일어나는지를 주 단위로 해부해 보자. 4주 견적의 웹 기능 프로젝트.

1주차: 견적대로 설계가 끝난다. 여기까지 모든 것은 계획과 같다. 2주차: 코어 개발을 시작하고 서드파티 API 문서에 “이 기능은 샌드박스에서는 동작하지 않는다”는 문구를 발견한다. 실사andbox 계정을 신청하는데, 승인 기다리느라 나흘이 간다. [추정] 3주차: 개발은 70퍼센트 진전했다. 그런데 상대가 “그 기능, 실제 사용자는 이렇게 쓸 거야”라며 사용 전제를 바꾼다. 한 줄의 피드백처럼 보였지만 데이터 구조를 다시 짜야 할 수준이다. 4주차: 원래 계획의 끝. 실제로는 아직 통합이 끝나지 않았다. 5주차에서 7주차: 재통합, 상대의 첫 검토, 검토에서 나온 열 건의 변경, 그중 세 건은 범위였다. 8주차: 두 번째 검토. 9주차: 마무리와 배포.

이 해부에서 눈에 들어오는 건 세 가지다. 초과한 주수의 대부분은 “개발”이 아니라 “기다리기”와 “바꾸기”였다. 둘 다 간트 차트에는 칸이 없다. 그리고 2주차의 나흘 승인이 4주차를 밀었고, 4주차가 5주차를 밀었다. 초과는 한 곳에서 생기지 않는다. 한 곳에서 시작해 주를 따라 이동한다. 그래서 1주차에 버퍼가 없으면, 4주차에 무릎을 꿇는다. 이 구조를 이해하면 3장의 버퍼와 체크포인트가 왜 그 모양인지 알게 된다. ## 추정 오류는 과정의 문제

이 책의 주장은 이것이다. 당신의 추정이 낮은 이유는 성격이 아니라, 과정을 불완전하게 쓰고 있기 때문이다.

지금 쓰는 과정은 대개 세 단계다. 프로젝트를 생각하고 "나 생각엔 몇 주가 걸리나" 하고 묻고 그 숫자를 말한다. 문제가 두 번째 단계에 있다. 뇌에게 인사이드 뷰만 묻고 있기 때문이다. 뇌는 매끄러운 경로로 답한다.

이 책은 두 번째 단계를 바꾸는 것이다. 밖에서 보게 하고(레퍼런스 클래스, 2장), 버퍼를 명시하게 하고(3장), 일정대로 다시 맞추게 한다(3장). 그리고 그 과정을 개인 시스템으로 고치게 한다(4장). 장을 넘기면서 각 단계가 무엇을 하는지 하나씩 설치해 가는 구조다.

페이지를 넘기기 전에 작은 작업을 하나 하자. 최근 3개 프로젝트를 적어본다. 각각의 최초 견적과 실제 소요일. 그리고 초과한 이유 상위 3개. 이유를 세 갈래로 나눠 본다. 예측 가능했던 것, 예측 불가능했던 것, 상대나 외부 환경이 바꿔 버린 것. 대부분 후자 두 갈래에 끼어들어 가 있을 것이다. 바로 그 빈틈이 인사이드 뷰로는 보이지 않는 부분이며, 이 책의 나머지 장이 채우는 부분이다.

그리고 한 가지만 달라 보자. 주어다. "나는 추정이 안 된다"는 성격 진단이고 거기서 끝난다. "이번 견적은 검토 사이클을 빼먹어서 40퍼센트 낮았다"는 과정 진단이고, 고칠 수 있다. 이 책은 과정 진단을 만들기 위해 쓰였다.

그 연습 결과를 바로 읽는 법도 함께 알려주겠다. 세 프로젝트의 "예측 가능했던" 이유가 전체의 절반을 넘으면, 문제는 인식이다. 당신이 장애물을 알았는데도 계획에 넣지 않은 것이다. 이 경우 2장의 레퍼런스 클래스보다, 3장의 버퍼 명명부터 시작하라. 반대로 "예측 불가능"과 "상대가 바꾼" 이유가 대부분이라면, 문제는 인식의 범위다. 당신은 정직하게 계획을 세웠고 세울 수 없는 영역이 견적을 깎아먹은 것이다. 이 경우 2장의 밖에서 보기부터 시작하라. 어느 쪽이든, 다음 견적에서 바꿀 대상이 하나 정해지는 것 자체가 이 장의 목표다.

과정 교체가 얼마나 비싼지도 계산해 둘 가치가 있다. 레퍼런스 클래스를 찾고 버퍼를 이름 붙이고 체크포인트를 달리는 일. 견적마다 30분에서

1시간쯤 걸린다. 한 달에 다섯 건의 견적이면 두 시간 반이다. 그 두 시간 반이 평균 30퍼센트의 초과를 10퍼센트로 줄여준다면, 계산은 복잡할 필요가 없다. [추정] 이 책의 나머지 세 장은 그 두 시간 반을 어디에 쓰는지를 알려주는 내용이다.

2 밖에서 보는 숫자: 레퍼런스 클래스

인사이드 뷰는 "이 프로젝트는 얼마나 걸릴까"를 묻는다. 아웃사이드 뷰는 다른 질문을 한다. "이런 프로젝트는 보통 어떻게 끝났지?" 이 장은 후자를 배우는 장. 방법은 단순하다. 지금 추정 중인 프로젝트와 비슷한 과거 사례를 찾아, 실제 소용일을 나열하고 그 분포의 위에서 견적을 낸다. 레퍼런스 클래스 방식.

2.1 레퍼런스 클래스란

레퍼런스 클래스는 추정 중인 것과 비슷한 프로젝트들의 모임이다. 비슷하다는 건 몇 가지 축에서 비교 가능하다는 뜻이다. 규모(랜딩 페이지인지 풀 앱인지), 도메인(내부 도구인지 고객 상품인지), 팀(1인인지 소규모인지), 환경(요구가 안정적인지 빠른지). 이 모임이 필요한 이유는 "실제"를 모으기 위해서다. 과거 프로젝트의 견적이 3주였다는 건 쓸모없다. 실제로 6주가 걸렸다는 건 쓸모있다.

클래스 정의는 구체적으로 해 보자. 식당 앱의 "테이블 예약" 기능을 추정하고 있다고 하자. 클래스를 한 줄로 쓰면, "3년 차 서비스의 신규 기능, 규모 3주에서 6주, 1인 또는 2인 팀, 서드파티 API 포함"이다. 이 한 줄이 클래스다. 이제 이 한 줄에 맞는 사례를 모은다. 지난해 한 고객에게 한 예약 기능(견적 4주, 실제 7주), 이전 프로젝트의 쿠폰 시스템(견적 3주, 실제 5주), 동료 팀의 좌석 선택 기능(실제 9주). [추정] 완벽하게 같은 사례는 필요 없다. 축에서 70퍼센트 정도 겹치면 클래스. 나머지는 보정으로 다룬다.

가장 믿을 수 있는 클래스는 자기 자신의 기록이다. 본인이 끝낸 프로젝트, 본인이 만든 기능, 본인이 초과한 일. 3년 정도 이 일을 했다면 20건에서 50건의 쓸 만한 사례가 있을 가능성이 높다. [추정] 없거나 너무 오래됐으면

다음 소스로 확장한다. 같은 일을 하는 동료, 고객 쪽의 경험(상대가 과거에 한 프로젝트), 업계 자료. 소스가 멀어질수록 데이터는 거칠어진다. 하지만 거친 데이터가 없는 데이터보다 낫다.

소스를 고를 때 한 가지 규정이 있다. 클래스는 자기 견적을 보기 전에 정한다. 먼저 "이건 어떤 종류의 프로젝트지"를 정의하고, 그다음 사례를 찾는다. 견적이 나온 다음에 사례를 찾으면, 자기 숫자를 지지하는 사례만 고르게 되기 때문이다. 그건 더 이상 아웃사이드 뷰가 아니다, 인사이드 뷰의 사후 정당화에 불과하다.

2.2 소스의 순서와 주의점

자기 기록이 얇을 때, 소스는 순서가 있다.

첫째, 동료. 같은 일을 하는 사람에게 "그거 몇 주 걸렸어?"라고 묻는다. 동료의 답은 대개 실제 소용일이다. 동료는 견적을 묻지 않으니까. 다만 한 가지를 확인한다. 그 동료가 비슷한 조건이었는지. 팀 규모가 달랐거나, 상대가 달랐으면, 그건 다른 클래스의 사례다.

둘째, 고객의 과거. 상대가 이전에 같은 종류의 일을 했는지 묻는다. "지난번에 비슷한 기능 했을 때 몇 주 걸렸어요?" 고객의 답은 두 가지 가치를 가진다. 클래스 데이터가 되기도 하고 상대의 기대치도 드러난다. 상대가 "3주면 되지"라고 말하면, 그 3주는 상대 클래스의 최솟값. 그리고 최솟값은 1장에서 배운 대로 유혹이다.

셋째, 업계 자료. 주의가 가장 많이 필요한 소스다. 업계 조사는 자기보고가 대부분이고 사람들이 조사에서 말하는 숫자는 견적이지 실제가 아니다. 즉, 이미 낙관이 섞인 숫자. 업계 수치를 쓰려면 1장에서 본 낙관을 한 번 더 빼줘야 한다. "업계 평균은 6주"라는 자료가 있다면, 당신의 클래스 데이터로는 "6주가 넘을 가능성이 높다"는 신호 정도로만 쓰자.

2.3 숫자를 고르는 법

비슷한 사례 8개가 있다고 하자. 실제 소용일이 3주, 4주, 4주, 6주, 8주, 9주, 11주, 14주. 견적은 어디에서 내는가.

유혹은 두 개다. 최솟값인 3주에서 내거나, 중앙값인 6주에서 내기. 둘 다 같은 이유로 틀리다. 분포 위에서 초과한 프로젝트는 걸러야 할 예외가 아니다. 견적에 포함시켜야 할 실제 가능한 시나리오.

추천하는 방법은 이 분포의 75퍼센트 부근에서 견적내는 것이다. 예시에서는 9주에서 11주 사이. 9주라면 8건 중 6건은 기한 안에 끝내고, 2건은 초과한다. 11주라면 마지막 1건만 초과. 목표는 "높게 부르는 것"이 아니다. "어느 분위(네 분의 일)의 사례까지 초과를 허용할 것인가"를 명시적으로 고르는 것. 일을 잃는 게 끔찍하면 중앙값, 초과 대가가 크면 90퍼센트. 중요한 건 그 선택을 드러내는 것이지, 무의식적으로 최솟값을 취하는 것이 아니다.

두 시나리오로 연습해 보자. 시나리오 A: 자기 팀의 내부 도구. 늦으면 손해지만 벌금은 없다. 중앙값 7주로 견적한다. 시나리오 B: 지연 벌금 조항이 있는 계약 프로젝트, 그리고 소개로만 오는 유일한 클라이언트. 같은 8건의 데이터에서도, 여기서는 11주로 견적. 데이터가 변한 게 아니다. 초과할 때 지불하는 대가가 변한 것이다. 그래서 퍼센타일 선택의 질문은 "넘으면 무엇을 지불하나"다.

퍼센타일을 고를 때 은밀한 디테일이 하나 더 있다. 클래스가 불확실할수록(사례들이 서로 더 달랐을수록) 더 높은 퍼센타일에서 가져가야 한다. 클래스가 잘 맞았다고 확신하면 중앙값도 쓰인다. 확신이 없으면 위쪽이다. 클래스 선택 자체가 불안한데 낮은 퍼센타일로 견적을 내면, 그것이 이 추정에서 가장 위험한 결정이 된다.

2.4 과거 사례가 없을 때

가장 흔한 반박은 "이 프로젝트는 과거 사례가 얼마 없어요"다. 사실이다. 프로젝트마다 새 요소가 있다. 하지만 "데이터가 없다"와 "데이터가 약하다"는 다른 상황이다. 약한 데이터의 대응은 포기하는 게 아니라, 클래스를 넓히는 것이다.

예를 들어 배달 앱의 구독 결제 기능을 처음 만든다고 하자. "배달 앱의 구독 기능"이라는 클래스는 비어 있다. 그러면 넓힌다. "내가 과거에 만든 구독 결제 기능"으로. 그래도 부족하면 "결제 연동이 있었던 기능"으로. 올라갈수록 클래스는 거칠어진다. 거칠어질 때마다 보정을 붙인다.

보정을 숫자로 해 보자. 넓혀서 모은 "결제 연동 기능" 클래스의 실제는 10주, 12주, 14주, 15주, 18주. 그런데 이번 프로젝트의 범위는 그 사례들보다 작다. 얼마나 작나? 기능 개수로 보면 60퍼센트쯤 된다. 그러면 75퍼센트 부근(15주)에 60퍼센트를 곱한다. 약 9주. [추정] 이 9주는 "클래스가 넓어서 보정한" 숫자다. 중요한 건 보정을 명시하는 것, 즉 "범위가 클래스 평균의 60퍼센트라 60퍼센트 보정"이라는 문장을 견적서에 쓰는 것이다. 그 문장이 있으면 상대가 묻을 수 있고, 당신이 답할 수 있다. "대충 7주"라고만 하면 아무도 묻지 못하고, 초과 시점에 싸움이 된다.

넓혀도 빈 클래스면 고객에게 묻는다. 상대는 비슷한 프로젝트를 해봤을 가능성이 높다. 고객의 실제 소용일이 바로 데이터다. "혹시 비슷한 기능 과거에 하셨을 때 몇 주 걸리셨어요?"가 추정 단계에서 가장 좋은 아웃사이드 뷰 질문이다. 대부분 기억이 있고 그 기억이 곧 레퍼런스 클래스가 된다.

2.5 먼저 아웃사이드 숫자, 그다음 인사이드 뷰

견적을 세울 때 익숙한 생각이 든다. "아웃사이드 숫자는 9주인데, 이 프로젝트는 달라. 상대가 협조적이고 기술은 익숙하니까 더 빠를 거야." 이 생각을 어떻게 다룰 것인가.

답은 거르지 말고 검사하는 것이다. 먼저 묻는다. "인사이드 뷰가 어떤 부분에서 이긴다고 보는가?" 답이 "상대가 협조적이다"면, 한 발 더 간다. "과거 8건에서 협조적인 상대는 몇 건이었어?" 8건 중 5건이 협조적인 상대였는데, 그 5건은 4주에서 14주까지 흩어져 있었다면, "상대가 협조적이다"는 3주를 깎을 정당한 이유가 아니다. 그건 클래스 전체에 공통인 조건이다.

정당한 차이를 숫자로 테스트하는 방법은 간단하다. 조건을 클래스에서 빼고 남은 사례의 시간이 비슷한지 본다. "협조적인 상대"를 빼면 3건이 남는데, 그 3건도 5주에서 9주에 흩어져 있었다면, 그 조건은 기저다. 반대로 "이번 범위가 클래스 평균보다 40퍼센트 작다"면, 남은 사례와 지금 범위를 비교할 수 있다. 4주짜리 범위의 사례가 5주, 6주였다면, 이번 견적은 5주에서 6주 사이로 움직이는 게 정당하다. 이 검사는 5분이면 끝난다. 그런데 추정 과정에서 단위 시간당 회수가 가장 큰 단계다.

결국 인사이드 뷰가 이길 수 있는 건 진짜 차이를 가리킬 때뿐이다. "이번 범위가 클래스 평균보다 40퍼센트 작다", "이 특정 라이브러리에 2년 경험을 갖췄다". 이런 건 숫자를 움직인다. "이번엔 진지하다", "상대가 좋아"는 움직이지 않는다.

2.6 연습

지금 추정 중인 프로젝트 하나를 골라 이 장을 해본다. 먼저 레퍼런스 클래스를 한 줄로 정의한다. 어떤 종류의 프로젝트, 어떤 규모, 어떤 팀.

그다음 비슷한 사례 5건에서 10건과 실제 소용일을 적는다. 자기 기록이 우선이고 없으면 동료, 없으면 고객. 중앙값과 75퍼센트를 찾는다.

그다음 두 질문을 한다. "이 클래스는 견적을 보기 전에 정했는가", "인사이드 뷰가 진짜 차이로 이기는가". 첫 질문이 아니오면, 클래스를 다시 정의한다. 두 질문이 아니오면, 아웃사이드 숫자를 둔다. 예가 아니면, 차이를 문장으로 쓰고 보정을 한다. 이 과정이 만들어낸 숫자라면, 견적을 내는 자리가 달라진다. 그 숫자는 "방어할 수 있다"고 말하는 숫자다. 차이가 크다. 다음 장에서는 이 숫자 위에 버퍼와 재추정을 올린다.

2.7 레퍼런스 클래스를 쓰는 순간의 느낌

처음 클래스 숫자를 견적으로 쓰는 사람은 대부분 어색함을 느낀다. "이 숫자는 내가 산 게 아니라, 과거 사례가 산 숫자야"라는 감각. 그래서 무의식적으로 자기 쪽으로 당기려 한다. 9주가 7주가 되고 7주가 6주가 된다. 이 당김을 막는 장치가 하나 있다. 견적 옆에 클래스를 쓴다.

"9주. 근거: 비슷한 사례 8건의 75퍼센트. 사례는 [목록]." 이 한 줄이 당김을 막는다. 상대가 "6주는 아니야?"라고 물으면, 8건의 목록을 펴서 답할 수 있다. 목록이 없으면, 당신은 9주라는 숫자만 가지고 있다. 그리고 숫자만 있는 견적은 첫 번째 협상에서 무너진다.

클래스는 또 당신 자신을 위한 기록이기도 하다. 3개월 뒤, 이 프로젝트가 실제로 몇 주에 끝났는지 볼 때, 당신이 "9주로 봤지, 왜 그랬지"라고 묻는 순간에 답이 된다. 답은 8건의 사례다. 그 답이 있어야, 4장의 로그가 성립한다. 레퍼런스 클래스는 견적의 근거일 뿐 아니라, 다음 견적의 데이터가 되는 구조다.

2.8 레퍼런스 클래스가 말하지 않는 것

레퍼런스 클래스는 소용일의 분포를 말해줄 뿐, 원인을 말해주지 않는다. 비슷한 사례의 75퍼센트가 9주였다는 걸 알게 되도, 그 9주 중에 어느 주가 지연됐는지는 알 수 없다. 그 정보는 로그(4장)와 버퍼 명명(3장)이 담당한다. 이 구분이 중요하다. 숫자만 있고 원인이 없으면, 9주를 견적했을 때 그 9주는 "알 수 없는 9주"가 된다. 실제로 초과가 시작됐을 때, 어떤 주가 잠식되고 있는지 모른 채 막을 수 없다. 그래서 숫자와 원인을 함께 기록한다. "9주, 그리고 버퍼를 먹을 리스크는 검토 사이클과 벤더 승인 두 개"가 완성된 견적이다.

두 번째로, 클래스는 "시간"이지 "비용"이 아니다. 시간을 돈으로 바꾸려면 자기 단가에 곱한다. 이때 한 가지 규율이 있다. 버퍼를 메우려고 단가를 낮추지 않는 것. "9주에 통상 단가"가 견적이다. "7주에 단가 조금 깎아서"는 베팅이다. 깎인 것이 당신의 수익이 되기 때문이다.

3 버퍼를 넣고 다시 맞추다

이 장은 견적을 단단하게 만드는 도구 두 개를 다룬다. 출발 전에 넣는 버퍼, 진행 중에 고치는 재추정. 2장에서 나온 아웃사이드 뷰 숫자는 여전히 예측이다. 약속이 아니다. 버퍼와 재추정이 예측을 관리 가능한 것으로 바꾼다.

3.1 버퍼는 거짓 부풀리기가 아니다

많은 사람이 버퍼를 숫자 부풀리기로 생각한다. "실제로는 3주인데 4주라고 하는 것." 그 의미라면 버퍼는 거짓이다. 이 책이 말하는 버퍼는 다르다. 정확히 계산할 수 없는 리스크에 붙이는, 명시적 비용이다.

차이는 투명성과 위치에 있다. 거짓은 숫자 속에 숨는다. 버퍼는 숫자 옆에 놓인다. "기본 추정 6주, 버퍼 3주, 합계 9주"는 "9주"와 다른 문장이다. 전자는 상대가 구성을 이해하게 하고 협상하게 한다. 상대가 "버퍼 2주로 줄이자"고 하면, 받아들일지 말지를 결정한다. 받아들인다면, 어떤 리스크를 자신이 흡수하는지 명확히 알 수 있다. 숨은 부풀리기는 협상할 수 없다. 그저 불신으로 남는다.

두 번째 차이. 특정 리스크에 붙는 것이 버퍼다. "결제 대행사 승인 기다리는 동안 2주"는 버퍼다. "혹시 모를 것에 대비해 2주"는 버퍼가 아니다. 이름을 안 붙인 버퍼는 패딩이고 패딩은 현실에 부딪히면 사라진다. 첫 번째 돌발 사건에 소모되고 끝이다. 이름을 붙인 버퍼는 버티는 이유가 있다. 자신이 만들어졌던 리스크에 맞춰 소모되기 때문이다.

3.2 버퍼를 어디에 놓는가

어떤 리스크에 버퍼를 둘 것인가. 불확실성이 기준이다. 크지만 불확실성이 낮은 태스크에는 버퍼가 적게 필요하다. 작지만 불확실성이 높은 태스크에는 버퍼가 크게 필요하다.

보통의 웹 서비스 프로젝트를 하나 꺼내 보자. 디자인. 불확실성 낮음. 많이 해봤고 산출물이 명확하다. 버퍼 10퍼센트, 또는 없음. 코어 개발. 중간. 20퍼센트에서 30퍼센트. 외부 서비스 연동. 높음. 50퍼센트 이상. 상대의 검토 사이클. 매우 높음. 100퍼센트 이상. 검토 사이클은 가장 큰 숨은 소모자다. 버전을 보내면 상대가 일주일째 연락이 끊기고, 열 건의 변경 요구와 함께 돌아온다. 그중 두 건은 "사실 처음부터 이게 의도였어요"다. 거의 모든 프로젝트에서 일어난다. 그리고 인사이드 뷰에는 절대 없다.

크기를 정하는 간단한 방법이 하나 있다. 작업을 5개에서 10개 태스크로 쪼개고 각각에 대해 "가장 빨라면, 가장 느리다면"을 적는다. 그다음 스프레드를 본다. 빨라야 하는 숫자와 느려야 하는 숫자가 가까운 태스크는 불확실성이 낮으니, 버퍼는 작게. 두 숫자가 멀리 벌어진 태스크는 불확실성이 크니, 버퍼는 크게. 그리고 "가장 느리다면"을 쓸 때 규칙이 하나 있다. "생각할 수 있는 최악"은 너무 넓다. "실제로 일어난 최악"이 적당하다. 과거 사례에서 실제로 그렇게 느렸던 숫자를 쓰는 것. 최악은 필요 없고 실제로 일어난 최악이면 된다.

버퍼의 합계가 견적이 된다. 기본 추정 더하기 버퍼 합계. 기본 6주에 버퍼 합이 3주면, 9주를 부른다. 구성을 드러낸다. 끝. 그 위에 별도의 "안전마진"을 더하면, 거짓이다.

3.3 버퍼를 태스크 위에 올리는 실습

구체적으로 한 번 해 보자. 9주 견적 프로젝트의 태스크 5개.

첫째, 디자인. 중간값 1주, 빨라면 1주, 느리다면 2주. 버퍼 없음. 많이 해본 영역이니까. 둘째, 코어 개발. 중간값 2주, 빨라면 1.5주, 느리다면 3주. 버퍼 3일. 셋째, 외부 서비스 연동. 중간값 1.5주, 빨라면 1주, 느리다면 3주. 버퍼 1.5주. 스프레드가 가장 넓은 태스크다. 넷째, 내부 테스트. 중간값 0.5주, 빠르고 느림의 차이가 작다. 버퍼 2일. 다섯째, 상대 검토 2회. 중간값 1주, 빨라면 1주, 느리다면 2.5주. 버퍼 1주. 상대의 시간을 기다리는 태스크라서, 내가 아무리 빨리 보내도 버퍼는 줄지 않는다.

중간값을 합한 기본 추정: 6주. 버퍼 합계: 3주. 견적: 9주. [추정] 주목할 것은 버퍼의 분포다. 3주 중 2.5주가 연동과 검토, 즉 스프레드가 넓고 통제권이 없는 두 태스크에 집중돼 있다. 디자인과 테스트에는 5일뿐이다. 이게 "불확실성에 비례한 버퍼"의 실제 모습이다.

"느리다면"을 쓸 때 과거의 실제를 참조하는 것이 이 실습의 핵심이다. 그가 버퍼를 패딩과 구분 짓는 선이다. 연동 태스크의 "느리다면 4주"는 지난 3건의 연동에서 실제로 4주를 먹은 적이 있다는 기록에서 나온 숫자다. 기록이 없으면 동료가나 고객에게 묻는다. "느리다면"을 상상으로 채우는 순간, 버퍼는 다시 패딩이 된다.

3.4 버퍼의 소유권

버퍼는 당신의 소유다. 상대가 "1주 줄여줘"라고 할 때, 침묵하고 1주를 자르는 것은 프로의 동작이 아니다. 버퍼를 자르는 건 리스크를 빼는 것이다. 그래서 답은 메뉴다.

"1주를 줄이려면, 검토 사이클 버퍼를 1주로 줄여야 해. 그러면 검토가 한 번만 길어지면 마감도 그만큼 밀려. 이건 네가 감수하는 리스크야." 이 문장을 말하면, 협상이 리스크 싸움으로 바뀐다. 상대는 두 가지 중 하나를 고른다. 리스크를 자기가 떠안고 마감을 지키거나, 마감을 늘리고 리스크를 당신이 지거나. 둘 다 합법적이며 둘 다 명시적이다.

여기서 "할인"과 "위임"의 차이가 나온다. 할인은 당신이 리스크를 흡수하는 것. 위임은 상대가 리스크를 떠안는 것. 프로의 견적 협상은 위임 협상이다. 그리고 위임이 기록으로 남을 때, 나중에 싸움이 되지 않는다. "검토 버퍼 1주는 귀사에 위임했죠"라는 한 줄이, 3개월 후의 분쟁을 막는다.

3.5 진행률 80퍼센트는 남은 20퍼센트가 아니다

재추정의 가장 큰 적은 "거의 끝났다는 감각"이다. 프로젝트가 80퍼센트 완료됐을 때, 뇌는 "남은 20퍼센트"를 결론짓는다. 그런데 실제로는 그렇게 작동하지 않는다. 앞의 80퍼센트는 계획한 부분이다. 마지막 20퍼센트는 계획하지 않은 부분. 통합, 엣지 케이스, 상대의 조미 요청, 프로덕션에서만 드러나는 버그. 버퍼는 대개 정확히 이 마지막 20퍼센트에서 소모된다.

그래서 재추정은 "진행률 얼마냐"고 묻지 않아야 한다. "남은 작업이 무엇이고 각각 얼마나 걸리나"라고 묻는다. 남은 작업을 새로 나열한다. 기존 태스크 리스트를 재사용하지 않는다. 기존 리스트는 "거의 끝난" 태스크로 가득하고 거의 끝난 태스크가 가장 비싸다. 남은 일을 별도 리스트로 만들고, 각각에 대해 다시 아웃사이드 뷰를 쓴다. 남은 작업의 아웃사이드 뷰란, "이런 마무리 작업은 과거에 얼마나 걸렸나"다. 그리고 버퍼를 붙인다. 그 합이 새 견적이다.

새 견적에는 함정이 하나 있다. 원래 마감까지 남은 시간보다 클 때가 많다. 원래 9주에 6주가 흘렀다면, 남은 작업 견적이 5주라는 건 합계 11주라는 뜻이다. 이 지점에서 선택지는 두 개. 마감을 미루거나, 범위를 줄인다. 둘 다 합법적이다. 위법적인 선택지는 하나뿐. 아무것도 고르지 않은 채 더 열심히 하는 것. 열심히 하는 것은 견적을 바꾸지 않는다. 버퍼를 태우기만 하고 늦어졌다는 사실조차 모른 채 늦게 만들어낼 뿐이다.

3.6 체크포인트와 재추정 일정

언제 재추정할 것인가. 견적을 내는 시점에 미리 정한다. 9주 프로젝트라면 3주, 6주, 9주에 체크포인트. 또는 작업 진행의 25퍼센트, 50퍼센트, 75퍼센트 지점. 각각에서 앞에서 설명한 재추정을 한다. 한 가지 규칙 더. 새 견적이 옛 견적과 같아도 한다. 재추정은 정기 측정이다. 체중은 쪘다고 느껴질 때만 재는 법이 없듯이.

보고 규칙이 중요하다. 새 견적이 나빠졌다면, 체크포인트에서 이유와 함께 보고한다. "6주차부터 3주 더 걸릴 예정입니다. 원인은 지난 검토에서 나온 변경 사항, 그리고 대행사 승인 지연." 상대의 반응은 생각보다 작다. 미리 알았던 지연에는 놀란다. 납품일에야 발견한 지연에는 화를 낸다. 나쁜 소식의 크기와 보고 시점은 별개다. 일찍 보고하고 이유를 붙이고 선택지를 두 개 제시한다. 미루거나, 줄이거나. 이게 "지연했다"를 "리스크를 관리했다"로 바꾸는 커뮤니케이션이다.

3.7 재추정에서 자주 하는 실수

재추정 단계에서 반복되는 실수가 세 가지 있다.

첫째, 기존 태스크 리스트를 재사용한다. "남은 태스크가 3개니까 2주쯤 되겠지." 남은 태스크에 "거의 끝난" 항목이 들어 있고 그게 가장 비싸다. 거의 끝난 항목은 검증이 남았고, 검증은 계획에 없던 일이다. 남은 일은 새로 나열한다.

둘째, 마감에 앵커링한다. "금요일까지는 끝내야 하니까, 2주라고 하자." 마감은 소망이지 데이터가 아니다. 데이터는 남은 일의 실제 분량이고 금요일은 그 위에 겹치는 원망이다. 둘을 합쳐 숫자를 만들면, 자기 암시가 된다.

셋째, 침묵으로 미룬다. 새 견적이 나빠진 걸 알면서 말하지 않고 그냥 계속

일한다. 1주일 침묵, 2주일 침묵. 그리고 납품일에 "사실은 조금 늦어질 것 같아요"를 말한다. 그 순간 상대가 잃는 것은 당신에 대한 신뢰 전체다. 침묵한 1주는 보고한 1주보다 10배 비싸다. 체크포인트의 존재 이유는 바로 이 침묵을 제도적으로 끊기 위해서다.

3.8 체크포인트의 실제 한 회

6주차 체크포인트를 구체적으로 하나 써 보자. 원래 9주 견적의 프로젝트, 현재 6주가 지났다.

재추정 시작. 남은 일을 새로 나열한다. 외부 연동은 80퍼센트 끝났지만 상대 벤더의 스테이징이 밀려 검증이 안 된다. 내부 테스트는 아직 시작 전. 상대 검토도 시작 전. 남은 일: 연동 검증 2주(벤더 스테이징 지연 반영), 내부 테스트 1주, 검토 2주. 합계 5주. 원래 계획의 남은 시간은 3주였다. 편차: 2주.

보고 문장. "6주차부터 5주가 더 걸릴 것으로 추정합니다. 원인은 벤더 스테이징 지연과, 지난 검토에서 나온 변경 세 건. 옵션이 두 개입니다. (1) 마감을 2주 늘립니다. (2) 두 번째 검토를 출시 후로 돌리고, 원래 마감에 1차 버전을 냅니다."

상대는 (2)를 고른다. 프로젝트는 원래 마감에 1차 버전으로 도착한다. 두 번째 검토는 출시 후 2주 안에 처리된다. 이 결과가 6주차에 5분짜리 재추정과 3줄짜리 보고를 했기 때문이라는 점을 기억할 것. 체크포인트는 늦음을 관리 가능한 형태로 바꾸는 도구다.

3.9 마감이 고정된 프로젝트: 시간을 고정하고 일을 줄인다

마감을 못 옮기는 프로젝트가 있다. 행사가 있는 날 오픈해야 하는 사이트, 특정 날짜에 맞춰 출시해야 하는 기능. 이런 프로젝트에서는 버퍼 방식이 반전된다. 시간을 먼저 고정하고 그 안에 얼마나 들어가는지를 역산한다.

방법은 단순하다. 마감에서 전체 버퍼(이름 붙인 리스크들의 합)를 뺀다. 남은 시간이 "보장 작업 시간"이다. 범위를 그 시간에 맞춘다. 들어가지 않는 일은 "다음 단계" 리스트로 빼간다. 여기서 유혹은 "버퍼를 줄이자"다. 최악의 수. 마감이 고정됐다는 건 리스크가 현실이라는 뜻이다. 버퍼는 달력과 협상할 수 없다. 협상할 수 있는 건 범위다.

이 움직임의 완성은 최소 가능 버전이다. 핵심 기능만으로 마감에 도달하는 버전을 내고 나머지를 다음 단계로 둔다. 절충이 아니라 전문적인 움직임이다. "모든 것을 보장하지는 못하지만, 이것까지는 이 날짜까지 보장합니다"를 말하는 것. 상대는 그 답을 듣기 위해 견적을 의뢰한 경우가 많다.

3.10 연습

지금 진행 중인 프로젝트에서 세 가지를 한다. 첫째, 시간을 가장 많이 태울 리스크 3개를 이름 붙인다. 각각에 주차 단위로 버퍼를 배정한다. "조금"이 아니라 "2주"로. 둘째, 체크포인트 2개에서 3개를 달력에 넣는다. "언젠가 보자"가 아니라 "금요일 오후 3시에 재추정한다"로. 셋째, 다음 체크포인트에서 견적이 달라지면 무엇을 보고할지 문장을 미리 쓴다. 보고할 수 없어 주저하는 재추정은 이미 지연이다.

4 추정을 시스템으로: 개인 규율 만들기

이전 장들은 부품이었다. 아웃사이드 뷰, 버퍼, 재추정. 이 장은 부품을 바로 쓰는 시스템으로 조립한다. 그리고 이 책에서 가장 실용적인 질문, "얼마나 걸리나요?"에 어떻게 답하는지도 다룬다.

4.1 추정 로그

추정에서 가장 중요한 자산은 자기 자신의 실제 데이터다. 그것을 가장 싸게 모으는 방법이 로그다.

로그는 표 하나다. 추정하는 프로젝트마다 한 줄을 추가한다. 열은 적다. 프로젝트 이름, 견적 날짜, 견적(버퍼 포함), 실제 소용일, 쓴 레퍼런스 클래스, 편차 이유. 끝. 추정할 때 한 줄을 만들고 끝날 때 실제와 이유를 채운다. 편차 이유는 한 줄이면 된다. "대행사 승인이 예상보다 2주 걸림", "상대가 범위 절반으로 줄임".

로그의 힘이 나타나기까지는 10건에서 20건 정도가 필요하다. [추정] 그 뒤에 자기 패턴이 보인다. 외부 서비스 연동을 늘 부족하게 부르는구나, 검토 사이클은 늘 1.5배가구나, 이 종류 프로젝트의 기본 추정은 실제의 2배 정도였구나. 그 "2배"가 개인의 보정계수다. 앞으로 기본 추정치에 곱해 쓸 수 있다. 마법이 아니다. 자기 역사가 숫자로 된 것일 뿐.

로그를 쓸 때 규정이 하나 있다. 실제로 한 것만 쓴다. 머릿속으로는 9주, 입으로는 6주를 말했다면, 6주를 기록한다. 로그는 이상적인 나를 위한 게 아니라, 실제의 나를 위한 도구다. 이상적인 나를 기록하면 한 달 만에 중단된다.

두 번째 규정. 작아서 견적이라고 부르기 힘든 것도 기록한다. 2일짜리 작업, 1시간짜리 견적. 작은 사례는 편차가 적고 쌓이는 속도가 빠르다. 그리고

작은 것을 기록하는 습관이 큰 것을 기록하는 습관이 된다.

로그의 모양을 한 번 그려 보자. 세 줄이면 충분히 느낌이 온다.

첫 줄: "랜딩 페이지 / 견적 3주 / 실제 4주 / 클래스: 과거 랜딩 / 이유: 폰트 라이선스 승인 1주". 두 줄: "예약 기능 / 견적 6주 / 실제 11주 / 클래스: 유사 기능 6건 / 이유: 연동 2주, 검토 3주". 세 줄: "관리자 도구 / 견적 2주 / 실제 2주 / 클래스: 관리자 도구 / 이유: 없음". [추정]

세 줄 읽는 것만으로 패턴이 보인다. 큰 편차는 연동과 검토에서 나오고 자기 작업에서는 안 나온다. 12줄이 쌓이면, 이 패턴은 숫자가 된다.

보정계수 계산도 숫자로 해 보자. 12건 합산: 견적 총 78주, 실제 총 179주. 179를 78로 나누면 약 2.3. 당신의 보정계수는 2.3이다. [추정] 앞으로는 두 가지 용도가 있다. 첫째, 기본 추정치에 곱해서 sanity check: 4주짜리 기본이 2.3배면 9주가 되어야 하고 당신이 6주를 견적하려면 그 근거(클래스가 더 작다는 것 등)가 필요하다. 둘째, 클래스 수치를 볼 때: "75퍼센트가 9주"인데, 9주가 자기 계수 2.3을 이미 반영한 실제의 분포인지, 아직 반영 안 된 견적의 분포인지 구분한다. 둘은 완전히 다른 것들이다.

계수는 영구로 유지되지 않는다. 실력이 붙고 로그가 쌓이면, 2.3은 1.7로, 1.7은 1.3으로 이동한다. 10건마다 다시 계산한다. 계수가 내려가는 게 이상한 게 아니라, 시스템이 작동하는 증거다.

4.2 꾸준히 조금 높은 쪽으로

질문이 하나 있다. 목표는 정확성인가, 꾸준함인가. 답은 꾸준함이다. 그리고 "조금 높은 쪽"의 꾸준함.

이유는 1장에서 본 비대칭이다. 낮은 쪽으로 틀리는 대가가 크고 높은 쪽으로 틀리는 대가는 작다. 그래서 목표 함수는 "오차의 비용을 줄이는

것”이다. 그 비용을 줄이는 숫자는 조금 높은 숫자다. 레퍼런스 클래스의 75퍼센트에 10퍼센트에서 20퍼센트 여유를 더한 곳이 좋은 시작점이다. [추정]

꾸준히 조금 높게 부르는 견적에는 예상 밖 효과가 있다. 상대가 숫자를 믿기 시작한다. “그 사람이 6주라면 6주겠지.” 그 신뢰는 자본이다. 다음 견적이 높아도 묻지 않는다. 그 전까지 열 번이나 기한을 지켰기 때문이다. 반대로 세 번이나 늦었으면, 가장 낮은 숫자에조차 “정말?”이 붙는다.

여기에도 함정이 있다. “조금”이 “크게”로 바뀌면 안 된다. 2배씩 높게 부르는 견적은 전문적으로 보이지 않고, 일을 잃는다. 목표는 조금 높은 곳에 낙착하는 것, 그리고 시간이 지나면서 그 “조금”을 줄이는 것. 로그가 쌓이면 보정계수가 정교해지고, 견적과 실제의 간격이 좁아진다. 이 시스템의 끝은 “5퍼센트 안에서 높다”다.

4.3 견적 전 4단계 검사표

추정 과정을 검사표로 정리하자. 프로젝트를 견적할 때 이 4단계를 차례로 밟는다. 순서를 바꾸지 않고 건너뛰지 않는다.

첫 단계, 범위를 문서화한다. 한 페이지면 충분하다. 포함되는 것, 포함되지 않는 것, 검토 라운드 수. 이 단계의 목적은 문서가 아니다. “거의 정해진 것”을 실제로 정하게 만드는 것. 추정 오류의 대부분은 범위 모호성에서 온다. “회원 기능”은 범위다. “회원 관련 어떤 기능”은 범위가 아니다.

두 단계, 레퍼런스 클래스를 고른다. 한 줄 정의, 사례 5건에서 10건, 실제 소용일. 중앙값과 75퍼센트. 클래스는 자기 견적을 보기 전에 정한다.

세 단계, 태스크로 쪼개고 버퍼를 이름 붙인다. 5개에서 10개 태스크, 각각의 빨라면과 느리다면, 불확실한 태스크에 이름 붙인 버퍼.

네 단계, 체크포인트를 캘린더에 넣는다. 재추정 날짜, 달라지면 보고하는

문장.

총 30분에서 60분. [추정] 지금의 "일단 3주로 하고"보다 길다. 하지만 2배가 초과하는 평균의 과정과 비교하는 것. 60분은 보험이다.

한 프로젝트를 4단계로 돌려 보자. 피트니스 앱의 "리뷰 기능" 견적.

첫 단계, 범위. 한 페이지: 텍스트와 별점 리뷰, 리스트 페이지 포함, 리뷰 조정 관리 화면은 제외(다음 단계), 검토 2회. "리뷰 기능"이 "리뷰 텍스트, 별점, 리스트, 2회 검토"로 고정됐다. 두 단계, 클래스. "신규 리스트/디테일 작업, 3주에서 6주, 1인 팀, 외부 서비스 없음. 사례 6건: 3주, 4주, 4주, 5주, 7주, 8주. 중앙값 4.5, 75퍼센트 7. [추정] 세 단계, 태스크와 버퍼. 디자인 1, 개발 2, 리스트 1, 엣지 0.5. 기본 4.5주. 버퍼: 검토 사이클 1.5, 엣지 케이스 0.5. 합 2주. 네 단계, 체크포인트 3주, 6주.

최종: 기본 4.5 더하기 버퍼 2, 6.5. "7주, 검토 2회 포함"으로 견적. 클래스의 75퍼센트(7)와 계산(6.5)가 거의 일치하는 걸 봤을 때, 이 숫자는 괜찮은 신호다. 두 출처가 같은 곳에 낙착했다.

옵션으로 다섯 번째 단계를 하나 더 붙여도 좋다. 프리모템. 시작 전에, 이 프로젝트는 실패했다고 가정하고, 가장 그럴 만한 원인 3개를 쓴다. "상대의 목표가 중간에 바뀜", "중요 의존성이 출시를 늦춤", "핵심 개발자가 장기 이탈". 각 원인에 대해, 일찍 알 수 있는 신호와 대응을 적는다. 프리모템은 실패를 일찍, 그리고 문서로 알아채기 위한 것.

프리모템을 실제로 하나 채워 보자. 프로젝트: 10주 계약 앱. "10주에 실패했다고" 가정하고 가장 그럴 만한 원인 세 개.

첫째, 고객사의 브랜드 검토가 3주를 넘는다. 신호: 20일째에 검토 답변이 아직 없는 것. 대응: 대기 항목을 문서로 보내, "답변 대기 중 항목 7개"를 기록으로 남긴다. 둘째, 푸시 알림 서비스의 인증이 생각보다 오래 걸린다. 신호: 7일째에 인증 요청을 못 넣은 것. 대응: 인앱 알림 대체안을 개발 범위에 포함. 셋째, 1인 개발자의 장기 이탈(병, 개인 사정). 신호: 없음(이건

감지 불가). 대응: 문서화를 계속 해서, 대체자가 1주 안에 들어올 수 있게 둔다.

이 세 줄이 프리모템이다. 실패를 막지는 못한다. 실패를 일찍, 그리고 문서로 알아채게 한다. 재추정이 필요로 하는 건 정확히 그다.

4.4 '더 싸게'라고 했을 때

고객이 "더 싸게 안 돼?"라고 할 때, 숫자를 내리지 않는다. 메뉴를 연다.

"줄이는 방법이 세 개인데, 각자 대가가 있어. (1) 범위 줄이기: 리스트 페이지를 빼면 2주 줄어. (2) 버퍼 줄이기: 검토 사이클 버퍼를 1주로 줄이면 1주 줄어. 대신 검토가 길어지면 마감도 길어. (3) 정산 방식 바꾸기: 4주 고정, 초과분은 시간 단가."

세 옵션은 각각 리스크를 어디에 둘지를 정한다. (1)은 기능을 내린 것, (2)는 상대가 리스크를 떠안은 것, (3)는 초과 리스크가 상대에게 가는 것. 당신은 대가를 명시한 채로 선택을 상대에게 넘긴다. 그리고 그 선택을 기록한다. "2번 옵션으로 검토 버퍼 1주 귀사 위임"이라는 한 줄. 나중에 "왜 2주 줄어드는데 1주만 줄었어?"라는 질문에 답이 된다.

숫자만 내리는 협상은 항상 당신에게 불리하다. 내린 숫자에서 버퍼가 어디에서 깎였는지 모르기 때문이다. 메뉴로 협상하면, 깎인 것이 무엇인지, 그 대가를 누가 지는지 기록이 남는다.

4.5 "얼마나 걸리나요?"에 답하는 법

마지막으로, 가장 실용적인 질문. 상대가 "얼마나 걸리나요"라고 물으면, 어떻게 답하는가.

단일 숫자를 말하지 않는다. 범위와 조건을 말한다. "4주에서 7주입니다. 검토 2회 포함. 결제 연동이 빨리 되면 아래쪽." 이 답에는 장점이 세 개

있다.

첫째, 정직하다. 범위는 불확실성을 드러내고 단일 숫자는 숨긴다. 단일 숫자를 들은 상대는 당신이 확신하는 줄 안다. 그리고 그 확신으로 계획한다. 깨질 때 피해가 더 크다.

둘째, 부담이 아니다. 범위가 약해 보인다고 생각하기 쉽다. "4주에서 7주"는 "5주"보다 모호하니, 상대는 만족하지 않을 거라고. 그런데 실제로는 조건을 붙인 범위가 더 만족을 준다. 왜냐하면, 아래쪽인지 위쪽을 무엇의 조건이 정하는지가 보이기 때문이다. "대행사가 빨리 되면 아래쪽"은 정보다. "5주"는 정보가 아니다.

셋째, 협상이다. 범위는 상대에게 선택지를 준다. 확신을 원하면 위쪽을, 속도를 원하면 리스크를 감수하고 아래쪽을 고른다. 선택을 함께 하면, 유지도 함께 한다.

견적을 거절해야 하는 상황도 있다. 범위가 정해지지 않았을 때. "포함되는 게 아직 안 정해져서, 지금은 숫자를 드리기 어렵습니다"가 전문적인 답이다. 정해지지 않은 범위에 숫자를 강제로 붙이는 건, 초과 프로젝트의 시작이다. 일을 따르는 것보다, 초과하는 것을 피하라. 초과한 일은 잃은 일보다 비싸다.

4.6 오늘부터

이 책은 여기서 끝난다. 방법은 모두 나왔다. 아웃사이드 뷰, 버퍼, 재추정, 로그. 하지만 안 쓰는 방법은 방법이 아니다. 읽은 기억일 뿐.

그래서 과제는 하나만 준다. 셋이 아니다. 하나. 오늘, 문서를 열고 추정 로그를 만든다. 한 줄만 쓴다. 최근 끝난 프로젝트. 견적, 실제, 편차 이유. 3분의 작업.

그 한 줄에서 시스템이 시작된다. 다음 프로젝트의 견적은 4단계 검사표를

밟고 버퍼는 이름을 갖고, 체크포인트는 캘린더에 있다. 그리고 그다음 프로젝트의 실제가 로그의 다음 줄을 채운다. 열 줄이 쌓이면, 당신만의 숫자가 생길 것이다. 당신의 보정계수. 그 숫자가 이 책의 끝이자, 당신의 추정의 시작이다.

추정 실력은 재능이 아니다. 지능도 아니다. 같은 과정을 100번 같은 방식으로 한 기록이다. 그리고 그 기록은 오늘 3분에서 시작된다.