Malorean Corp.

공감할 수 있는 소설

이해를 돕는 파노라마

개발자, AI와 작은 기업: 변화에 관한 (거의) 실화

오늘날 AI를 마주한
SaaS ERP 팀들이
실제로 겪는 일



LAURENT DESÉCHALLIERS

Malorean Corp

> 어제까지만 해도 하나의 아이디어를 프로토타입으로 만드는 데 몇 주가 걸리곤 했습니다. 오늘날에는 인공지능의 도움을 받는 한 사람이 며칠 만에 해낼 수 있습니다. 놀라운 변화입니다. 하지만 진정한 격변은 다른 곳, 바로 우리가 함께 일하는 방식에서 일어날지도 모릅니다.

Laurent Deséchalliers

번역에 관한 안내

이 판본은 프랑스어 원문을 바탕으로 인공지능이 자동 번역했습니다.

인간 번역가의 검수를 거치지 않았습니다. 아이디어와 구조, 예시는 원문에 충실하지만, 일부 표현이 어색하거나 몇몇 대목에 원문인 프랑스어의 흔적이 남아 있을 수 있습니다.

이는 이 책을 프랑스어로만 남겨 두지 않고 여러 언어로 제공하기 위해 감수한 대가입니다. 표현이 불분명할 때는 프랑스어판을 기준으로 삼아 주십시오.

들어가며

AI를 보며 제가 가장 놀라는 것은 코드를 작성하는 능력이 아닙니다. 얼마 전까지만 해도 며칠, 때로는 몇 주가 걸렸을 일을 가능하게 만드는 속도입니다.

실현 가능성에 관한 논의가 끝나기도 전에 아이디어가 프로토타입으로 바뀔 수 있습니다. 개발자는 팀 전체가 준비될 때까지 기다리지 않고도 한 가지 가능성을 탐색할 수 있습니다. 프로그래밍을 모르는 사람도 자신이 구상한 것의 첫 번째 버전을 직접 만들 수 있을 때가 있습니다. 그렇다고 프로토타입이 곧 신뢰할 수 있는 제품이 되는 것도, 복잡성이 사라지는 것도 아닙니다. 하지만 직접 시도해 볼 수 있다는 사실만으로도 무엇이 가능한지 바라보는 관점은 이미 달라집니다.

그러나 우리 기업은 첫 번째 해결책을 만들어 내는 일이 이토록 빨라질 수 있는 세상을 전제로 설계되지 않았습니다.

기업은 업무를 진전시키는 데 필요한 전문성, 가용 시간, 업무 인계와 여러 판단을 중심으로 조직되어 왔습니다. 이런 방식에는 나름의 이유가 있습니다. 하지만 개인이 발휘할 수 있는 역량이 달라지면 새로운 질문이 생기기 시작합니다. 여전히 같은 순서로 일해야 할까요? 같은 승인을 기다려야 할까요? 같은 업무를 계속 같은 직무에만 맡겨야 할까요?

이 질문은 개발자에게만 해당하지 않습니다. 프로덕트 매니저, 디자이너, 영업 담당자, 팀 리더를 비롯한 많은 사람도 새로운 업무 방식을 발견하고 있습니다. 이전에는 동료에게 요청하던 일의 일부를 이제 직접 해낼 수 있는 사람도 있습니다. 자신이 하던 일의 성격 자체가 달라지는 사람도 있습니다. 도구의 문제처럼 보였던 변화가 우리가 협업하고 의사결정을 내리는 방식에 영향을 미치기 시작했습니다.

결말까지 이미 알고 있는 혁명이 시작되었다고 선언하고 싶을 수도 있습니다. 하지만 저는 아직 그 단계에 이르렀다고 생각하지 않습니다.

어떤 기업은 업무와 책임을 새롭게 나누는 방식을 실험하고 있습니다. 또 다른 기업은 때로는 충분한 이유를 가지고 더 신중하게 움직입니다. 유망해 보이는 방식도 있지만, 일상의 제약 속에서도 살아남을지 판단하기에는 아직 경험이 부족합니다. 작은 팀에 적합한 해법이 더 크고, 규제가 많고, 역사가 긴 기업에서도 자동으로 유효해지는 것은 아닙니다.

그래서 저는 AI 시대에 따라야 할 올바른 조직 형태를 설명하는 지침서를 쓰려 하지 않았습니다. 이 책은 실제로 무엇이 움직이고 있는지 이해하려는 시도입니다. 도구가 무엇을 가능하게 하는지, 기업은 어떻게 반응하는지, 무엇을 얻고 무엇을 잃을 위험이 있는지, 그리고 어떤 질문에 답해야 하는지를 살펴봅니다.

이를 위해 서로 교차하는 두 갈래 길을 택했습니다.

첫 번째는 관찰입니다. 실제 기업과 공개된 업무 방식, 연구와 실험을 살펴볼 것입니다. 확립된 사실과 아직 가설에 머무는 것을 구분하고, 매력적인 발표를 입증된 결과로 오해하지 않으려 합니다.

두 번째는 하나의 이야기입니다. 가상의 기업 Malorean Corp.를 따라가며 직원들이 새로운 가능성이 자신의 일에 어떤 변화를 가져오는지 차츰 발견하는 과정을 지켜볼 것입니다. 이들에게 조직을 재설계할 계획은 없습니다. 대신 고객과 습관, 긴급한 일, 신념, 때로는 의견 충돌이 있습니다. 허구는 아무것도 증명하지 않습니다. 다만 사람의 눈높이에서 이 변화를 바라볼 수 있게 해 줍니다.

두 관점은 모두 필요합니다. 실제 사례는 우리의 직감을 사실로 착각하지 않도록 도와줍니다. 이야기는 조직의 변화가 조직도 안에서만 일어나지 않는다는 점을 일깨웁니다. 변화는 신뢰와 역량, 책임, 그리고 각자의 자리에도 영향을 미칩니다.

결국 앞으로 이어질 모든 페이지를 관통하는 질문은 하나입니다. 기업을 이루는 사람들이 자신의 직무가 지금까지 허용했던 것보다 훨씬 더 많

은 일을 할 수 있다는 사실을 발견하면, 그 기업에서는 어떤 일이 벌어질까요?

저자 소개

저는 수십 년 동안 소프트웨어 산업에서 일해 왔고, 현재는 프로덕트 매니저로 일합니다. 개발과 아키텍처에서 시작해 팀 운영으로, 다시 제품과 그 운영, 마케팅으로 영역을 넓혀 왔습니다.

이 경험 덕분에 저는 인공지능의 발전과 그것이 우리의 일에 미치는 영향을 가까이에서 지켜볼 수 있습니다. 일상적인 업무 속에서 직접 변화를 경험하는 한편, 동료 프로덕트 매니저들과의 대화를 통해서도 많은 것을 배웁니다. 우리는 경험과 발견, 의문을 서로 나눕니다. 이런 대화는 종종 제시선을 도구 너머로 이끕니다. 우리의 직무와 의사결정, 함께 일하는 방식에서 실제로 달라지는 것은 무엇일까요?

관찰과 대화가 쌓이면서 질문도 늘어났습니다. 한 걸음 물러서서 우리의 인상을 다른 기업에서 벌어지는 일과 비교하고, 그 생각을 나누고 싶었습니다. 그렇게 이 책이 탄생했습니다.

글쓰기는 오래전부터 제 경력과 함께해 왔습니다. 전문 언론에 기고했고 M2 Édition과 Eyrolles에서 기술 감수자로 일했습니다. 지금은 malorean.net에서 프로덕트 매니지먼트, AI, 일하는 방법, 정보 시스템에 관해 글을 씁니다.

전략 및 실무 마케팅을 다루는 독립 잡지 *Proposition Marketing*에도 기고하고 있습니다. 마케팅과 기업 경영 분야의 독자를 위해 글을 쓰다 보면 공통의 용어를 안다고 가정하지 않고 제품에 관한 문제를 이해하기 쉽게 설명하게 됩니다.

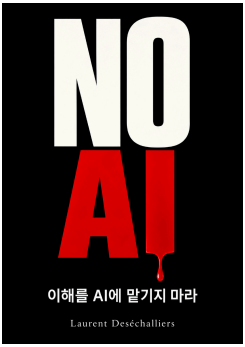
Laurent Deséchalliers

- laurent@malorean.net
- https://malorean.net

같은 저자의 책



Le Contexte — AI가 중요한 것을 이해하고, 당신의 지식을 연결하며, 배운 것을 목표를 위해 축적할 수 있게 해 주는 컨텍스트를 만드는 방법.
출간: *2026*년 *10*월



NO AI — 이해를 *AI*에 맡기지 마라 기억을 가꾸고, 사고를 깊게 하고, 비판적 사고를 단련하기 위한 AI 없는 방법. 우리 자신의 역량을 키우고, 이해를 AI에 맡기지 않으면서 AI를 더 잘 활용하기 위한 실천.
출간 예정: *2026*년 말

자세한 내용: https://malorean.net.

감사의 말

먼저 정기적으로 의견을 나누는 프로덕트 매니저 여러분께 감사드립니다. 우리는 프로덕트 매니지먼트와 개발 직군의 변화, 더 넓게는 인공지능이 일과 기업 조직에 미치는 영향에 관해 서로의 관점을 견줍니다. 때로는 열띤 이 대화가 끊임없이 새로운 질문과 반성, 아이디어를 불러옵니다. 그중 일부는 이 책의 여러 페이지에도 담겨 있습니다. 여러분의 호기심과 공유해 주신 경험, 토론하려는 열의에 감사드립니다.

그리고 제 아내 Christine에게 감사하고 싶습니다. 첫 책 *Le Contexte*를 마치며 잠시 쉬겠다고 약속했지만, 저는 거의 곧바로 이 책을 시작했습니다. 다행히 두 번째 여정은 첫 번째만큼 우리의 저녁과 주말을 많이 빼앗지는 않았습니다. 제 생각이 또 한 권의 책으로 이어질 때에도 인내하며 응원해 준 Christine에게 고맙습니다.

예시에 관한 안내

이 책의 생각과 예시는 30년이 넘는 IT 업계 경험에서 나왔습니다. 서로 다른 프로젝트와 업무 환경에서 겪은 상황, 그리고 다른 회사의 많은 프로덕트 매니저, CTO, 엔지니어링 매니저 등과 나눈 대화가 바탕이 되었습니다. 여러 조직과 제 커리어의 여러 시기에서 쌓인 경험입니다.

책의 논지를 위해 이 상황들을 수정하고, 일반화하거나, 여러 사례를 재구성했습니다. 사용 사례, 수치, 지표는 반복해서 나타나는 문제와 그로부터 얻은 교훈을 보여 주기 위한 구성입니다. 실제 고객, 제품, 조직을 묘사하지 않으며, 제 현직장의 데이터, 업무 방식, 입장을 반영하지도 않습니다.

이 책에 담긴 분석, 방법, 견해는 제 개인의 것이며 오직 제가 책임집니다.

이 책은 AI로 썼을까요?

그렇습니다. 전 과정을 인공지능과 협업해 썼습니다.

오늘날 이 질문이 특별한 의미를 갖는다는 것을 알고 있습니다. 사람들은 기사나 소설, 책이 AI로 쓰였는지 묻습니다. 어떤 이들은 인간에게서 나온 부분과 기계에서 나온 부분을 반드시 가려내야 한다는 듯 글을 탐지기에 넣어 보기도 합니다.

저는 솔직하게 밝히고자 합니다. AI는 자료 조사부터 집필에 이르기까지, 아이디어 구성, 참고 자료 검증, 미처 보지 못한 관점 탐색 등 이 책의 모든 단계에 참여했습니다. 하지만 이 협업만으로 누가 책을 생각했는지가 설명되지는 않기에, 조금 더 설명할 필요가 있습니다.

과거에는 이 정도 규모의 작업을 하려면 자료를 찾고, 아이디어를 검토하고, 원고를 읽고, 구조를 잡거나 형식을 다듬는 사람들과 함께할 수 있었습니다. 오늘날 AI는 그런 역량의 일부를 제게 제공합니다. 빠르게 실행하고, 표현을 제안하고, 모순을 짚어 내며, 때로는 제가 생각하지 못한 질문을 던집니다. 제가 가진 시간 안에서 혼자서는 다룰 수 없을 만큼 더 넓게 탐색하도록 도와줍니다.

이 책을 쓰기 위해 저는 ChatGPT 기술을 기반으로 한 컨텍스트 엔진을 사용했고, 정보의 지속성을 확보하는 데 Notion을 활용했습니다. 덕분에 자료와 아이디어, 결정, 작업 중인 여러 버전을 보존하고 다음 대화에서 다시 찾을 수 있습니다. 이 접근법은 첫 책 *Le Contexte*에서 이미 소개했습니다. 그 책에서는 이런 시스템을 통해 AI가 일종의 매우 유능한 개인 비서가 되는 과정을 설명합니다. AI는 질문에 답하는 데 그치지 않고 프로젝트의 흐름을 기억하며 장기간에 걸쳐 앞으로 나아가도록 돕습니다.

하지만 반드시 구분해야 할 것이 있습니다. 실행할 수 있는 능력과 이해하고 선택할 책임은 서로 다릅니다. 이 책을 이끄는 질문, 그 질문의 바탕이 된 직업적 경험, 해석과 판단, 그리고 무엇을 옹호하고 무엇을 제외할지에 관한 최종 결정은 제 몫입니다. AI는 제 생각에 이의를 제기하고, 비판하며, 더 명확하게 표현하도록 도울 수 있습니다. 하지만 저의 판단과 저자로서의 책임을 대신하지는 않습니다.

저는 또한 이 책과, 글쓰기 자체가 창작의 핵심인 작품을 구분합니다. François-Victor Hugo가 번역한 Shakespeare를 읽을 때든, Cervantès나 Cizia Zykë의 원문을 읽을 때든, 목소리와 리듬, 문체는 이야기의 내용만큼 중요합니다. 번역 작품에서는 번역가의 선택 또한 독서 경험의 일부가 됩니다. 이런 문학에서 AI에 어떤 자리를 내줄 것인지는 별도의 질문이며, 여기서 답을 내리려 하지는 않습니다.

이 책에는 허구가 일부 포함되어 있지만, 가장 중요한 목표는 기술적·조직적 변화를 이해하는 것입니다. 형식은 즐겁고 명료해야 하지만, 무엇보다 제가 나누고자 하는 생각과 사실, 질문을 전달하는 데 기여해야 합니다.

AI 덕분에 저는 본업 밖에서, 혼자 힘만으로는 아마 도달하기 어려웠을 규모로 이 프로젝트를 실현할 수 있었습니다. AI는 이 책과 같은 개인 프로젝트를 발전시키면서도, 그에 필요한 사고의 엄밀함을 포기하지 않도록 해 줍니다.

돌이켜 보면 AI가 일으키는 변화를 다룬 책을 바로 이런 방식으로 썼다는 데에는 제법 자연스러운 일관성이 있습니다.

Laurent Deséchalliers

목차

번역에 관한 안내 3

들어가며 4

저자 소개 7

같은 저자의 책 8

감사의 말 9

예시에 관한 안내 10

이 책은 AI로 썼을까요? 11

Malorean Corp. 15

1 Malorean Corp. — 평소와 다른 하루 16
2 AI는 개발자의 생산 능력을 비약적으로 키웠다 22
3 Malorean Corp. — 그다음 날 30
4 새로운 조직 모델이 등장하고 있다 42
5 Malorean Corp. — 할 수 있는 일과 해야 하는 일 50
6 역할의 경계는 흐려져도 책임은 남는다 63
7 Malorean Corp. — Alice가 글로 남기지 않고도 알고 있던 것 72
8 더 많은 자율성에는 더 적은 틀이 아니라 더 많은 틀이 필요하다 88
9 Malorean Corp. — 합리적인 두 결정 101
10 AI는 기업의 중간 계층을 사라지게 할까? 114
11 Malorean Corp. — 되찾은 시간 127
12 맺음말 143

Malorean Corp.

1 – Malorean Corp. – 평소와 다른 하루

Malorean Corp.의 본사는 르아브르에 있었습니다. 하지만 코로나19 이후로는 그 주소만으로 실제 업무가 이루어지는 장소를 설명하기 어려워졌습니다. 직원들은 프랑스 전역에 흩어져 있었습니다. 카메라 앞에서 만나고, 공유 화면을 보며 이야기했으며, 대부분은 같은 방에 있지 않아도 일을 진전시킬 수 있었습니다.

Malorean Corp.는 온라인 ERP를 개발했습니다. 기업의 거의 모든 업무에 조금씩 스며드는 그런 소프트웨어였습니다. 고객 중에는 직원이 몇 명뿐이고 요구사항도 매우 구체적인 영세 기업이 있었습니다. 복잡한 프로세스와 특별한 요구사항을 갖고 있으며 어설픈 대응을 좀처럼 용납하지 않는 대기업도 있었습니다.

고객은 대부분 프랑스 기업이었습니다. 독일과 스페인에도 사용자가 일부 있었지만, 주로 프랑스에서 이미 고객인 그룹의 현지 법인이었습니다. 최고경영자 Hubert Félix Duca는 더 먼 곳을 바라보고 있었습니다. 그는 Malorean Corp.가 유럽에서 본격적으로 성장하기를 바랐습니다.

문제는 ERP가 홍보용 웹사이트처럼 쉽게 국경을 넘지 못한다는 점이었습니다. 나라마다 규정이 달랐습니다. 급여 처리 관행과 행정 의무, 기업의 업무 방식도 달랐습니다. Hubert는 이런 제약을 잘 알고 있었습니다. 그는 소프트웨어 개발이 아니라 경영과 관리, 회계, 재무 분야 출신이었습니다. 그에게 기능은 고객이 곤란을 겪지 않고 사용할 수 있을 때에만 가치가 있었습니다.

회사 안에서 각자의 역할은 비교적 명확했습니다.

각각 개발자 다섯 명으로 구성된 두 팀이 소프트웨어를 발전시켰습니다. 두 프로덕트 매니저 Alice와 Bob은 이들과 함께 요구사항을 이해하고, 유용한 개선 사항을 선택하며, 제품의 방향을 정했습니다. Alice는 개발자 출신이어서 흐름을 놓치지 않고 기술 논의에 참여할 수 있었습니다. Bob은

경영대학원 출신으로 다른 방식으로 문제에 접근했습니다. 배경은 달랐지만 두 사람은 매우 잘 협력했습니다.

영업팀은 신규 고객을 발굴하고 기존 고객과의 관계를 관리했으며, 마케팅 담당자 한 명이 이를 지원했습니다. 전문가 다섯 명은 사전 영업 단계와 구축 과정에서 현장이나 원격으로 고객을 도왔습니다. 이들은 ERP를 설치하는 데 그치지 않았습니다. 기업이 요구사항을 구체화하고 프로세스를 조정하도록 도왔으며, 판매 후 6개월 동안 계속 지원하는 경우도 많았습니다.

고객지원팀에는 열두 명이 있었습니다. 일곱 명은 소규모 고객을, 다섯 명은 대규모 고객을 담당했습니다. 어떤 기능이 고객에게 소프트웨어를 제대로 쓰게 하기보다 우회 방법을 찾도록 강요한다는 사실을 가장 먼저 듣는 사람도 흔히 이들이었습니다. 법무 담당자 두 명은 Malorean Corp.의 법률 업무를 맡는 동시에 제품 설계에도 전문성을 보탰습니다. 특히 급여 규정처럼 민감한 규칙과 관련된 기능을 변경할 때 그 역할이 중요했습니다.

이 모든 것이 회사를 움직이게 했습니다. 물론 완벽하지는 않았습니다. 많은 고객이 사용하는 소프트웨어에는 요구사항과 예외가 쌓입니다. 바꾸기 어려운 습관도 생기고, 다시 만들 시간이 없어 그대로 두는 기능도 늘어납니다. 그래도 모두가 자신이 해야 할 일을 대체로 알고 있었습니다.

그러던 중 ChatGPT가 대화에 등장하기 시작했습니다.

처음부터 거창한 혁신 프로그램이 있었던 것은 아닙니다. 호기심이 많았던 두 사람, 영업팀의 Julie Morel과 개발팀의 Lucas Martin이 몇 달 전부터 사용해 보았습니다. 각자 자신의 업무에 쓰기 시작했고, 무엇을 해낼 수 있었는지 동료들에게 들려주었습니다.

Julie에게는 효과가 분명했습니다. 까다로운 이메일을 준비하면서 더 명확한 표현을 요청했고, 미팅 전에 메모를 정리했으며, 여러 아이디어를 발표 구성안으로 바꾸었습니다. 때로는 슬라이드 몇 장에 들어갈 내용도 준비했습니다. 고객에게 무엇을 말할지 결정한 것은 ChatGPT가 아니었습니

다. Julie가 재료를 제공하고, 제안을 다시 다듬고, 표현을 고친 뒤 자신이 책임질 수 있는 내용만 보냈습니다.

그녀는 도구가 자신보다 고객을 더 잘 안다고 주장하지 않았습니다. 이미 알고 있는 내용을 정리하는 데 30분씩 쓰지 않아도 된다는 사실이 그저 반가웠습니다.

동료 Philippe Garnier는 멀리서 이를 지켜봤습니다. 대기업 고객에 익숙한 노련한 영업사원인 그는 계약서 뒤에 있는 사람들을 알았습니다. 어떤 회의록에도 남지 않은 의견 충돌과, 회의의 의미를 바꾸어 놓는 짧은 말들도 알고 있었습니다.

— 이메일 쓸 때 한번 써 봐요. 형식만 다듬는 데라도요. 어느 날 아침 Julie가 말했습니다.

— 내 이메일은 내가 쓸 줄 알아요.

악의가 담긴 말은 아니었습니다. 그는 자신의 가치가 바로 고객에 대한 이해와 고객에게 말하는 방식에 있다고 생각했기에, 왜 업무의 일부를 도구에 맡겨야 하는지 알지 못했을 뿐입니다.

Julie는 더 권하지 않았습니다. 준비해야 할 미팅이 있었습니다.

반면 Lucas는 ChatGPT를 전혀 다른 방식으로 사용했습니다. 리옹에서 일하는 숙련된 개발자인 그는 반복 업무에서 몇 분을 아끼는 데만 관심을 두지 않았습니다. 많은 시간을 실험에 썼고, 갈수록 까다로운 질문을 던졌으며, 답이 마음에 들지 않으면 처음부터 다시 시도했습니다. Julie가 주로 쓸모 있는 결과를 얻으려 했다면, Lucas는 어디까지 갈 수 있는지를 탐색했습니다.

초기 사용자들의 경험은 마침내 Hubert에게도 전해졌습니다. 한 사람당 매달 20유로 남짓이라면 계산은 단순해 보였습니다. 모두가 번거로운 업무에서 조금이라도 시간을 되찾을 수 있다면 투자 가치는 충분했습니다. 그래서 Malorean Corp.는 전 직원에게 ChatGPT를 제공하기로 했습니다.

수많은 발표와 함께 혁명을 선언한 것은 아니었습니다. 필요하다고 느끼는 사람은 누구나 사용할 수 있었습니다. 매일 여는 사람도 있었고 드물게 쓰는 사람도 있었으며, Philippe는 전혀 쓰지 않았습니다. 대부분에게 AI는 여전히 보조 도구였습니다. 원하는 것을 설명하면 AI가 제안하고, 사람이 확인한 뒤 결정했습니다.

회사의 일상은 계속되었습니다.

수요일, 제품팀과 개발팀의 논의에서 오래된 문제가 다시 등장했습니다. 휴가 관리 기능이었습니다.

누구도 이 모듈을 그다지 좋아하지 않았습니다. 오래되고 경직되어 있었으며 선택할 수 있는 기능도 많지 않았습니다. 계속 사용할 수 있을 만큼은 작동했지만, 관련 불만이 멈출 만큼 잘 작동하지는 않았습니다.

휴가를 신청하려는 직원은 업무용 컴퓨터를 사용해야 했습니다. 일요일 저녁에 휴가 생각이 났을 때 휴대전화로 간단히 처리할 방법이 없었습니다. 휴가 일수 계산에는 더 난처한 결함도 있었습니다. 모듈이 공휴일을 제대로 반영하지 못했습니다. 월요일부터 금요일까지 휴가를 신청했는데 수요일이 공휴일이면 나흘이 아니라 닷새를 차감할 수 있었습니다.

고객지원팀도 문제를 알고 있었습니다. 영업팀도 마찬가지였습니다. 일부 고객에게 이 모듈은 반복되는 불만거리가 되었습니다. 기대에 제대로 부응해서가 아니라, 이미 존재한다는 이유로 계속 사용하고 있었습니다.

— 제대로 다시 만들어야겠네요. 누군가가 정리하듯 말했습니다.

반대하는 사람은 없었습니다. 하지만 '제대로 다시 만든다'는 것은 화면을 재설계하고, 계산 규칙을 처리하고, 모바일에서도 사용할 수 있게 만들며, 시간이 지나 낡아 버린 부분을 교체한다는 뜻이었습니다.

두 달 정도 걸릴 작업으로 예상되었습니다. 다른 우선순위도 있었습니다. 이 문제는 또다시 언젠가 해야 할 일의 목록으로 돌아갔습니다.

금요일 오후, 화상회의가 끝날 무렵 Lucas가 몇 분만 시간을 달라고 했습니다.

— 보여 드릴 게 있습니다.

그는 주제를 미리 알리지 않았습니다. 수요일 논의 이후 그가 휴가 모듈을 작업했다는 사실을 아무도 몰랐습니다.

회의 화면에 그의 화면이 나타났습니다. 명료하고 현대적인 인터페이스가 열렸고, 필요한 곳에 핵심 정보가 놓여 있었습니다.

Lucas는 휴가 신청 방법을 보여 주었습니다. 그런 다음 창을 휴대전화 너비로 줄였습니다. 화면이 그에 맞게 바뀌었습니다. 데스크톱 컴퓨터로 돌아가지 않고도 계속 진행해 날짜를 고르고 신청을 확정할 수 있었습니다.

그는 두 번째 사례를 보여 주었습니다. 월요일부터 금요일까지 한 주를 신청하되 수요일이 공휴일인 경우였습니다.

차감 일수는 나흘로 표시되었습니다.

날짜를 바꾸자 결과가 다시 계산되었습니다. 모바일 화면으로 돌아가 몇 개의 화면을 스크롤하며 연이어 조작했습니다. 고정된 시안 몇 장을 차례로 보여 주는 것이 아니었습니다. 양식은 입력에 반응했고, 계산이 이루어졌으며, 전체 과정이 실제로 작동했습니다.

처음에는 동료 몇 명이 평범한 질문을 했습니다. 특정 옵션은 어디에 있는지, 날짜를 바꾸면 어떻게 되는지 물었습니다. 그러다 질문이 멈췄습니다.

화상회의의 작은 창들 속에서 사람들의 표정이 달라졌습니다.

수요일에는 두 달이 걸릴 일이라고 이야기했습니다. 금요일에 Lucas는 작동하는 첫 번째 버전을 보여 주고 있었습니다. 누구에게도 도움을 요청하지 않았습니다. 아무에게도 알리지 않았습니다. 그리고 아직 어떻게 해냈는지 설명하지도 않았습니다.

— 잠깐만요. 누군가 마침내 입을 열었습니다. 그거 그저께 이야기한 모듈 맞죠?

Lucas가 그렇다고 답했습니다.

침묵이 흘렀습니다.

화면 속 모든 사람이 같은 것을 바라보았습니다. 몇 주에 걸쳐 계획해야 한다고 생각했던 일이 이틀 만에 나타나 있었습니다.

어떻게 가능한 일이었는지 아는 사람은 아무도 없었습니다.

2 – AI는 개발자의 생산 능력을 비약적으로 키웠다

수십 년 동안 소프트웨어 개발 조직은 한 가지 단순한 제약을 중심으로 만들어졌습니다. 소프트웨어를 만드는 데에는 많은 비용이 듭니다.

아이디어를 제품으로 바꾸려면 시간과 희소한 전문성, 수많은 조율이 필요했습니다. 하나의 기능을 범위화하고, 작은 단위로 나누고, 우선순위를 정하고, 개발하고, 테스트하고, 검토하고, 배포해야 했습니다. 투입할 수 있는 개발자의 수와 전문성, 한 주제에 쓸 수 있는 시간이 기업이 시도할 수 있는 일의 범위를 자연스럽게 제한했습니다.

이 희소성이 조직을 빚었습니다. 기업은 개발자의 시간을 보호하고, 개발에 앞서 업무를 준비하며, 방해를 줄이고, 백로그를 구성하고, 실제 구축에 들어가기 오래전부터 선택과 조정을 하는 법을 배웠습니다. 현대적인 제품 조직의 운영 방식도 상당 부분 이런 환경에서 발전했습니다. 비용이 많이 드는 엔지니어링 역량을 잘못된 아이디어에 쓰지 않아야 했습니다.

코딩 도우미에 이어 더 폭넓은 작업을 수행할 수 있는 에이전트가 등장하면서 이 공식이 달라지기 시작했습니다.

더 나은 자동 완성만의 변화가 아니다

초기의 개발자용 AI 도구는 새로운 세대의 생산성 도구로 이해할 수 있었습니다. 함수를 완성하고, 코드 일부를 설명하고, 테스트를 작성하거나 리팩터링을 제안했습니다.

다음 세대는 일의 성격을 크게 바꿉니다. 에이전트는 작업을 전달받아 코드베이스를 살펴보고, 여러 파일을 수정하고, 테스트를 실행하고, 일부 오류를 고친 뒤 검토할 수 있는 상태로 변경 사항을 내놓을 수 있습니다. 따라서 개발자는 실행 과정의 상당 부분을 위임할 수 있으며, 무엇보다 예전보다 훨씬 많은 일을 동시에 움직이게 할 수 있습니다.

그 결과는 단순히 '개발자가 코드를 더 빨리 작성한다'는 데 그치지 않습니다.

한 사람이 시작하고 진전시킬 수 있는 소프트웨어 작업의 양이 크게 늘어납니다.

일부 기업은 이미 놀라운 결과를 공개하기 시작했습니다. 자사가 AI 네이티브 엔지니어링 시스템으로 전환한 과정을 설명한 AMPECO는 개발 속도가 네 배 빨라지고 스토리당 버그가 50% 줄었다고 보고했습니다. 이 수치를 시장 평균으로 보아서는 안 됩니다. 특정한 환경에서 한 기업이 발표한 결과이기 때문입니다. 하지만 목표로 하는 변화의 규모는 보여 줍니다. 이제는 함수 하나를 작성하는 데 몇 분을 아끼는 문제가 아니라 팀의 실행 능력을 크게 높이는 문제입니다. [1]

한 기업이 발표한 이 놀라운 결과만이 유일한 단서는 아닙니다. 저도 Claude Code를 사용하며 직접 경험했습니다. 나흘 동안 180회가 넘는 커밋을 거쳐 화면과 API, 테스트를 갖춘 제품의 첫 버전을 만들었습니다. 예전 방식이었다면 한두 달이 걸렸을 일이라고 생각합니다. 이 경험은 2026년 7월 Malorean.net에 소개했습니다. 물론 이 사례만으로 생산성 향상을 일반화할 수는 없습니다. 개발 경험이 있는 프로덕트 매니저가 이제 무엇을 할 수 있는지 보여 주는 동시에, 생성된 코드의 품질과 유지보수성, 통제에 따르는 어려움도 드러냅니다. [7]

이 지점에서 질문의 규모가 달라집니다.

AI가 코딩 시간을 몇 퍼센트 줄이는 데 그쳤다면 조직은 그대로 유지된 채 그 이익을 흡수할 수 있었을 것입니다. 생산 능력이 그보다 훨씬 빠르게 늘어나면 다른 제약의 상대적 중요성이 커집니다.

코드가 빨라진다고 기업 전체가 빨라지는 것은 아니다

소프트웨어 팀은 코드 작성이라는 고리 하나로만 이루어진 사슬이 아닙니다.

구축에 앞서 문제를 이해하고, 투자할 가치가 있는지 결정하고, 방향을 선택하며, 몇 가지 제약을 명확히 해야 합니다. 구축한 뒤에는 검토하고, 테스트하고, 보안을 확인하고, 배포하고, 관찰하고, 유지보수해야 합니다. 그 사이 내내 계속 판단하고 선택해야 합니다.

이 가운데 한 고리만 크게 빨라지면 다른 고리로 더 많은 일이 밀려갈 뿐입니다.

코드 생산은 빨라져도 고객을 이해하는 속도가 같은 비율로 빨라지지는 않습니다. 프로토타입은 급증해도 그중 무엇을 계속 발전시킬지 결정하는 능력까지 늘어나는 것은 아닙니다. 검토해야 할 변경 사항은 많아져도 이를 살필 사람의 주의력이 같은 비율로 늘어나지는 않습니다.

따라서 AI가 다른 활동을 갑자기 쓸모없게 만드는 것이 문제는 아닙니다.

AI는 병목 지점을 옮깁니다.

과거에는 실행이 느려 가려져 있던 요소들이 더 잘 보이게 됩니다. 선택한 문제의 질, 의도의 명확성, 판단과 승인, 일관성, 출시 후 학습하는 능력입니다.

더 많이 만드는 것이 더 많은 가치를 만드는 것은 아니다

이 구분이 핵심이 됩니다.

무언가를 만들기 어려웠을 때는 출시한 소프트웨어의 양이 조직의 가치 창출 능력과 비슷해 보일 수 있었습니다. 그때도 정확한 등식은 아니었습니다. AI 시대에는 위험한 착각이 됩니다.

팀은 제품을 개선하지 않고도 더 많은 코드와 풀 리퀘스트, 프로토타입, 기능을 만들어 낼 수 있습니다. 늘어난 물량이 결함과 재작업, 비용, 유지보수 복잡성을 키운다면 전체 성과가 오히려 나빠질 수도 있습니다.

다시 말해,

만들 수 있는 능력과 무엇을 만들 가치가 있는지 선택하는 능력은 다릅니다.

그래서 활동량을 나타내는 지표만으로는 더욱 부족합니다. 변경 건수나 코드의 양, 단순한 AI 도구 사용 여부는 팀이 더 많은 가치를 만들었는지 말해 주지 않습니다.

Canva는 이처럼 활동량에서 영향으로 관점을 옮긴 흥미로운 사례입니다. 이 회사는 제품·엔지니어링·데이터 팀이 실험의 상업적 영향을 추정하고, 현장의 의사결정과 사업 성과를 더 직접 연결할 수 있는 시스템을 개발했습니다. [2]

그러면 질문의 성격이 달라집니다.

이전에는,

우리는 얼마나 많이 만들 수 있는가**?**

이제는,

이토록 많은 것을 만들 수 있게 된 지금**,** 실제로 무엇을 만들 가치가 있는가**?**

> 소프트웨어 생산성 문제는 ***40***년이 넘었다
>
> *1986*년 *Fred Brooks*는 *1987*년 **IEEE Computer**에 실린 **No Silver Bullet**을 발표했습니다. 그는 소프트웨어 제작의 우발적 어려움, 즉 언어와 도구, 설계를 코드로 옮기는 문제와, 소프트웨어의 본질적 복잡성, 즉 시스템을 이해하고 설계하고 연결하며 발전시키는 문제를 구분했습니다. 그에 따르면 어떤 단일한 기술적·관리적 진보도 *10*년 안에 개발의 생산성과 신뢰성, 단순성을 열 배 높이겠다고 약속할 수 없습니다. *[3]*
>
> 이 글은 *1975*년에 처음 출간된 **The Mythical Man-Month**의 *1995*년 기념판에 다시 수록되었습니다. *[4]* 생성형 *AI*는 실행의 규모를 바꿀 가능성이 있지만, 선택한 문제가 옳은지까지 자동으로 보장하지는 않습니다. 이는 *Brooks*의 논의를 오늘의 상황에 비추어 본 것이며, 그가 ***AI*** 에이전트에 관해 내놓은 예측이 아닙니다.

새로운 병목은 판단이다

실행 비용이 낮아질수록 다른 자원의 상대적 가치는 커집니다. 고객에 대한 지식, 문제에 대한 이해, 전략적 맥락, 명확한 의도를 표현하는 능력, 위험 평가, 제품 감각, 거절하는 능력, 의사결정에 대한 책임입니다.

2026년 3월 Malorean.net에 발표한 글에서 저는 기술 도구의 접근성이 높아질수록 경쟁 우위가 맥락에 대한 이해로 이동한다고 주장했습니다. 여기에는 업무 영역과 사용자, 고객, 기술적 제약, 경쟁, 규제가 포함됩니다. 이는 실험 결과가 아니라 전략적 분석입니다. 판단력을 발휘한다는 말의 의미를 구체화합니다. 단순히 만들 줄 아는 것을 넘어, 무엇을 만드는 것이 옳은지 선택할 만큼 상황을 잘 이해하는 것입니다. [8]

AI는 '어떻게'를 더 쉽게 만듭니다.

그렇다고 '왜'까지 저절로 분명해지지는 않습니다.

조직이 문제를 다 이해하기 전에 해결책을 만들 수 있게 합니다. 기업의 시장과 맞지 않아 거절했어야 할 고객 요청도 빠르게 구현할 수 있게 합니다. 어떤 것이 실제로 더 나은 결과를 내는지 모른 채 한 기능의 여러 변형을 쏟아 낼 수도 있습니다.

만들기가 쉬워질수록 만들지 않을 것을 선택하는 규율이 더 중요해집니다.

Ramp는 이 원칙을 잘 보여 줍니다. 이 회사의 CPO는 우선순위가 높은 ICP와 특히 잘 해결하려는 문제를 명확히 해야 하며, 제품을 다른 방향으로 끌고 가는 요청이라면 수익성이 있더라도 거절해야 한다고 강조합니다. [5]

2026년 7월 Marty Cagan은 이 간극을 *AI 생산성의 역설*이라고 불렀습니다. 더 빨리 만드는 능력만으로는 애초에 만들 가치가 없던 아이디어의 문제가 해결되지 않는다는 진단입니다. 그는 학습을 위해 만드는 것과 성과를 내기 위해 출시하는 것을 구분합니다. 이는 제품 실무자의 분석이지, 모든 기업이 똑같은 방식으로 영향을 받는다는 것을 입증하는 측정 결과는 아닙니다. [6]

따라서 AI의 역설은 다음과 같습니다.

AI는 우리의 행동 능력을 크게 높이며, 그 결과 어떤 행동을 선택할지가 전보다 더 중요해집니다.

도구만이 아니라 시스템의 변화

모든 개발자에게 코딩 도우미를 나눠 주고 팀에는 예전처럼 계속 일하라고 하는 것으로 이 변화에 대응할 수도 있습니다.

그러면 국지적인 이익은 생깁니다.

하지만 그 이익이 커질수록 조직은 다른 한계에 부딪힙니다. 이해하고, 결정하고, 조율하고, 검증하며, 만들어진 것을 흡수하는 능력의 한계입니다.

이 단계에 이르면 더 이상 도구 도입만의 문제가 아닙니다.

조직의 문제가 됩니다.

엔지니어링이 가장 희소한 자원이던 때에 설계된 조직이 그 희소성이 줄어든 뒤에도 최적일 이유는 없습니다.

그러면 피할 수 없는 질문이 생깁니다.

희소한 자원, 즉 엔지니어링 역량을 보호하기 위해 설계된 조직은 그 역량이 훨씬 풍부해지면 어떻게 달라지는가?

참고 문헌

[1] How we built an AI-native engineering system – AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- 에이전트를 중심으로 엔지니어링 조직을 재구축한 경험과, 회사가 발표한 큰 폭의 역량 향상을 설명합니다.

[2] Measuring Commercial Impact at Scale – Canva

- URL: https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- Canva가 제품팀과 엔지니어링팀의 실험을 누구나 확인할 수 있는 상업적 영향 추정치와 연결하는 방법을 설명합니다.

[3] No Silver Bullet: Essence and Accidents of Software Engineering – Fred Brooks

- URL: https://www.cgl.ucsf.edu/Outreach/pc204/NoSilverBullet
- 1986년에 발표되고 1987년에 출간된 글로, 소프트웨어 구축의 본질적 복잡성과 우발적 어려움을 구분합니다.

[4] The Mythical Man-Month, 기념판, 16장 – Fred Brooks

- URL: https://www.oreilly.com/library/view/mythical-man-month-the/0201835959/ch16.xhtml
- 1995년 판에 *No Silver Bullet*이 수록되었음을 보여 주는 출판 목차이며, 장의 일부만 볼 수 있습니다.

[5] 5 insights from Ramp's CPO on building for hypergrowth – Ramp

- URL: https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- Ramp의 CPO가 집중, 우선순위 ICP, 방향을 흐트러뜨리는 요청을 거절해야 할 필요성을 강조합니다.

[6] The AI Productivity Paradox – Marty Cagan, Silicon Valley Product Group (2026)

- URL: https://www.svpg.com/the-ai-productivity-paradox/
- AI가 제공하는 속도 향상과 실제로 창출된 성과를 구분한 2026년 7월의 분석입니다. 인과관계를 입증한 연구가 아니라 실무자의 견해입니다.

[7] Quatre jours de vibe coding dans la peau d'un PM rouillé – Laurent Deséchalliers, Malorean.net (2026)

- URL: https://malorean.net/articles/2026-07-28-quatre-jours-de-vibe-coding-dans-la-peau-d-un-pm-rouille.html
- Claude Code를 사용한 개인적 경험으로, 제품 구축 속도와 기술적 어려움, 테스트, 품질 관리 문제를 다룹니다. 통제된 연구도, 일반화할 수 있는 증거도 아닙니다.

[8] Dans le logiciel, l'avantage ne sera plus la technologie. Ce sera la compréhension du contexte – Laurent Deséchalliers, Malorean.net (2026)

- URL: https://malorean.net/articles/2026-03-23-dans-le-logiciel-lavantage-ne-sera-plus-la-technologie-ce-sera-la-comprehension-du-contexte.html
- 희소성이 고객과 업무, 규제, 기술적 맥락에 대한 이해로 이동한다는 저자의 전략적 글이며, 정량적 입증 자료는 아닙니다.

3 – Malorean Corp. – 그다음 날

— AI를 사용했습니다.

Lucas는 모두가 방금 본 것을 설명하기에 그 말이면 충분하다는 듯 아주 자연스럽게 말했습니다.

그는 여전히 화면을 공유하고 있었습니다. 새 휴가 모듈이 창 대부분을 차지했습니다. 몇 분 전에는 휴대전화에서 휴가를 신청하는 방법과 소프트웨어가 휴가 일수를 계산할 때 공휴일을 반영하는 모습을 보여 주었습니다.

— 잠깐만요. Alice가 다시 물었습니다. AI를 사용했다는 게 코드 작성을 도와줬다는 뜻인가요?

— 네. 아니, 그것만은 아닙니다. 모듈을 설계하고, 기존 코드를 이해하고, 화면을 개발하고, 계산 기능과 테스트를 작성하는 데 썼어요. 사실상 거의 모든 일에 사용했습니다.

— 그걸 수요일부터 한 거예요?

— 수요일 오후부터요. 마지막 작업은 오늘 아침에 끝냈습니다.

Alice는 화면 속 얼굴들을 살펴보았습니다. 몇몇은 미소를 짓고 있었습니다. 다른 사람들은 자신들이 무엇을 놓쳤는지 찾는 듯했습니다.

— 하지만 명세가 없었잖아요. Alice가 지적했습니다.

— 네. 수요일에 이야기했고, 저는 그 모듈과 문제점을 알고 있었습니다. 먼저 AI에게 기존 구현을 분석해 달라고 했어요. 그다음부터는 하나씩 진행했습니다.

— 전부 작동하나요?

Lucas가 가볍게 웃었습니다.

— 전부인지는 모르겠어요. 방금 보여 드린 것은 작동합니다. 확인할 부분은 분명 더 있을 겁니다.

같은 팀에서 일하는 Thomas가 처음으로 끼어들었습니다.

— 그 뒤의 코드는 어떤가요?

— 직접 살펴보세요. 테스트도 꽤 많이 작성했습니다.

— 살펴볼게요. Thomas가 답했습니다.

그는 특별히 감탄한 것처럼 보이지 않았습니다. 아니면 감탄했지만 내색하고 싶지 않았는지도 모릅니다.

Hubert Félix Duca는 그때까지 말없이 있었습니다.

그는 수많은 시연을 지켜보았습니다. 10분 동안 잘 작동한 소프트웨어가 수백 개 기업에서 곧바로 사용할 수 있는 소프트웨어는 아니라는 사실을 알고 있었습니다. 예외와 대기업 고객의 특수한 요구사항, 실제 고객 환경에 들어가야 비로소 나타나는 문제도 알고 있었습니다.

하지만 수요일에 들은 말도 기억하고 있었습니다.

두 달.

그런데 지금 눈앞에는 당시 상상했던 것과 이미 상당히 비슷한 무언가가 있었습니다.

— Lucas, 이것에 시간을 얼마나 썼는지 다시 말해 주겠어요? Hubert가 물었습니다.

— 대략 이틀입니다.

Hubert는 화면 쪽으로 몸을 조금 기울였습니다.

— 알겠습니다.

그는 더 할 말을 찾지 못했습니다.

회의는 몇 분 뒤 끝났습니다. 모두 자신의 업무로 돌아갔지만, 그날 오후에 해야 할 일을 제대로 다시 이야기하는 사람은 없었습니다.

토요일 아침, Alice는 커피를 준비하며 시연을 다시 떠올렸습니다.

그녀를 불편하게 만든 것은 Lucas의 속도만이 아니었습니다. 누구의 도움도 필요하지 않았다는 사실이었습니다.

Alice는 프로덕트 매니저라는 자신의 일을 잘 알고 있었습니다. 개발을 시작하기 전에 문제를 이해하려 했고, 여러 요청을 비교했으며, 고객과 대화하고, 개발자들과 함께 합리적인 해법을 찾았습니다.

휴가 모듈 같은 주제라면 아마 여러 차례 논의를 마련했을 것입니다. 규칙을 구체화하고, 특수한 사례를 판단한 뒤 무엇부터 개발할지 결정하도록 도왔을 것입니다.

Lucas는 그런 일을 하나도 하지 않았습니다.

논의를 듣고, 스스로 시작하고, 해결책을 만들었습니다.

Alice의 한쪽 마음은 감탄했습니다. 다른 한쪽에서는 모든 개발자가 이런 방식으로 일하기 시작하면 어떤 일이 벌어질지 궁금해했습니다.

하지만 지난 몇 달 동안 그녀 자신의 일상도 크게 달라졌습니다.

처음에 Alice는 회의록을 요약하거나 문서를 다시 쓰는 데 AI를 사용했습니다. 그러다 더 중요한 작업도 맡기기 시작했습니다. 고객 피드백 처리의 일부, 몇몇 요약 자료 작성, 유사한 요청의 분류, 예전에는 몇 시간씩 걸리던 여러 행정 작업을 자동화했습니다.

과거에는 오전의 상당 부분을 차지하던 일을 이제 몇 분 만에 끝내기도 했습니다.

그 일들이 사라진 것을 아쉬워하지 않았습니다.

오히려 그 반대였습니다.

마침내 고객과 이야기하고, 요청 뒤에 숨어 있는 문제를 이해하고, 실제로 무엇을 만들어야 하는지 생각할 시간이 더 많아졌습니다.

AI는 그녀의 일을 덜 흥미롭게 만들지 않았습니다.

오히려 일을 제대로 하지 못하게 막던 일부 부담을 덜어 주었습니다.

그렇다면 개발자들만 예전과 똑같은 방식으로 계속 일해야 할 이유가 있을까요?

Alice는 잔을 내려놓았습니다.

질문은 너무나 분명해 보였습니다.

답은 훨씬 덜 분명했습니다.

Hubert 역시 주말의 상당 부분을 생각하며 보냈습니다.

그의 회사에는 늘 실현할 역량보다 아이디어가 더 많았습니다. 몇 달째 기다리는 기능이 있었고, 고객이 요청한 개선 사항 중 일부는 번번이 뒤로 밀렸습니다. 소프트웨어에 아직 많은 작업이 필요한 유럽 진출 계획도 있었습니다.

그는 장애물을 너무도 잘 알고 있었습니다. 모든 일을 할 시간이 부족했습니다.

어쩌면 금요일에 Lucas는 그 장애물이 더 이상 그렇게 넘기 어려운 것만은 아니라는 사실을 보여 주었을지도 모릅니다.

두 달로 예상했던 작업이 이틀 만에 첫 버전으로 나올 수 있다면, 갑자기 가능해지는 프로젝트는 얼마나 많을까요?

Malorean Corp.에는 개발자 열 명이 있었습니다. Hubert는 모두가 그만큼 빠르게 일할 수 있다면 어떤 일이 생길지 상상해 보았습니다.

그런 다음 개발자 열 명이 아무런 조율 없이 각자 기능을 만들기 시작하는 모습도 상상했습니다.

두 번째 장면은 훨씬 마음에 들지 않았습니다.

토요일에 곧바로 Alice에게 전화할까 생각했지만 참았습니다.

일요일 저녁에는 짧은 메시지만 보냈습니다.

내일 아침에 시간 있어요? 금요일 일에 대해 다시 이야기하고 싶습니다.

몇 분 뒤 답이 왔습니다.

네. 저도요.

월요일 아침 9시, Alice와 Hubert는 화상회의에서 만났습니다.

— 그래서요? Hubert가 물었습니다.

— 뭐가요?

— 금요일에 무슨 일이 있었는지 이해했어요?

Alice가 미소 지었습니다.

— 기술적으로는 대략요. Lucas는 AI를 사용해 우리가 상상했던 것보다 훨씬 빠르게 개발했습니다.

— 그 밖의 부분은요?

— 바로 그게 문제입니다.

Hubert가 고개를 끄덕였습니다.

— 주말 내내 생각했습니다. 우리는 몇 년째 역량이 부족해요. 요청을 거절하고, 프로젝트를 미루고, 하지 않을 일을 고르는 데 늘 시간을 씁니다. 그런데 Lucas는 두 달로 예상한 것을 이틀 만에 내놓았어요.

— 첫 번째 버전이에요. Alice가 강조했습니다. 아직 실제 운영에 투입할 준비가 됐는지는 모릅니다.

— 알아요. 하지만 완성하는 데 2주가 더 걸린다고 해도 많은 것이 달라집니다.

Alice도 반박할 수 없었습니다.

— 마음에 걸리는 것은, Hubert가 말을 이었습니다. 그가 Alice를 거치지 않고 모든 일을 했다는 겁니다.

Alice는 바로 답하지 않았습니다.

— 저도 그게 마음에 걸렸어요. 그녀가 인정했습니다.

— 우리가 일하는 방식을 다시 알려 줘야 한다고 생각해요?

— 저도 그렇게 생각해 봤습니다.

— 그런데요?

— 하지만 그건 실수일 것 같아요.

Hubert가 눈살을 찌푸렸습니다.

— 그래도 자신이 무엇을 개발할지 혼자 결정하기 시작하면 뜻밖의 문제가 생길 수 있습니다.

— 물론이죠. 하지만 실제로 일어난 일을 보세요. 수요일에 우리는 휴가 모듈을 다시 만들어야 한다고 말했습니다. 평소 방식대로 했다면 제가 요구사항을 수집하고, 회의를 잡고, 주제를 준비하고, 작업을 나누기 시작했을 겁니다. 무엇이든 보여 주기까지 아마 몇 주가 걸렸겠죠.

— 꼭 낭비한 시간이라고 볼 수는 없어요.

— 맞아요. 요구사항을 이해하는 일을 그만두자는 뜻도 아닙니다. 하지만 Lucas는 그 모든 일을 기다리지 않고도 진전할 수 있다는 것을 보여 줬어요.

— 그러면 개발자들이 스스로 결정하게 두자는 겁니까?

— 아니요, 오히려 그 반대예요.

Alice는 몇 초 동안 생각했습니다.

— 우리가 서로 다른 두 가지를 섞고 있는 것 같아요. 무엇을 해야 할지 결정하는 것과 어떻게 할지 결정하는 일입니다.

— 둘은 연결되어 있잖아요.

— 맞습니다. 하지만 그렇다고 모든 결정에 제가 관여해야 하는 것은 아니에요.

그녀는 자세를 조금 바로잡았습니다.

— 지난 몇 달 사이 제 일도 크게 달라졌어요. 저는 매일 AI를 사용합니다. 손으로 하던 수많은 일을 자동화했고, 문서를 준비하거나 정보를 정리하는 데 쓰는 시간도 훨씬 줄었습니다.

— 알아요. 이야기해 줬잖아요.

— 하지만 제 일이 얼마나 달라졌는지는 모르실 수도 있어요. 지금은 정말 중요한 일, 즉 고객을 이해하고 문제를 찾아내며 우리의 에너지를 어디에 쓸지 결정할 시간이 훨씬 많습니다.

Hubert는 듣고 있었습니다.

— 예전에는 조직 안에서 일이 진전되도록 하는 것도 제 업무의 일부였습니다. 티켓을 준비하고, 요청을 구체화하고, 질문에 답하고, 논의를 조율했죠. 앞으로 개발자들이 이런 부분에서 저를 훨씬 덜 필요로 하게 된다면요?

— 시간이 더 생기겠죠.

— 맞아요. 개발자들에게도요.

그녀는 잠시 말을 멈췄습니다.

— 우리는 AI가 제 일을 바꾸는 것은 받아들입니다. 그런데 왜 개발자의 일은 그대로여야 한다고 생각할까요?

Hubert는 의자 등받이에 몸을 기댔습니다.

— 알겠습니다. 그래도 무엇을 만들지는 누군가 결정해야 해요.

— 당연하죠. 무엇보다 왜 만드는지도요.

— 그러면 어떻게 해야 한다고 봐요?

Alice는 망설였습니다.

준비한 발표 자료는 없었습니다. 제안할 조직 모델도, 도입할 새로운 프로세스도 없었습니다.

직감 하나만 있었습니다.

— 뭔가를 한번 시험해 보면 어떨까 싶습니다.

Hubert는 그녀가 계속 말하기를 기다렸습니다.

— Lucas에게 한 가지 주제를 맡길 수 있어요. 실제 고객이 겪는 문제를요. 그리고 해결 방법을 찾는 데 훨씬 큰 자율성을 주는 겁니다.

— 명세 없이요?

— 미리 만든 해법을 주지 않는 거죠. 문제를 설명하고, 그 문제를 아는 사람들에게 접근할 수 있게 한 다음, 스스로 탐색하게 합니다.

— Alice는요?

— 계속 지원합니다. 요구사항을 이해하고, 아이디어를 검토하며, 단순히 만들 수 있다는 이유만으로 무언가를 개발하지 않도록 도울 겁니다. 하지만 모든 단계를 지시하지는 않아요.

Hubert는 생각했습니다.

— 그러면 프로덕트 매니저의 업무 일부를 맡기겠다는 건가요?

— 어쩌면요. 아니면 그저 우리 사이에서 일을 늘 기계적으로 나누는 것을 그만두는 것일 수도 있습니다.

— 잘되지 않으면요?

— 그때 알게 되겠죠.

Hubert가 미소 지었습니다.

— 방법이라고 하기에는 조금 허술하네요.

— 알아요.

Alice도 미소 지었습니다.

— 하지만 새로운 조직을 석 달 동안 논의하기보다 제한된 주제로 직접 시험해 보는 편이 낫습니다.

Hubert는 바로 대답하지 않았습니다.

그러다 무언가를 떠올린 듯했습니다.

— 마침 몇 주 전부터 일부 식품업 고객에게서 반복해서 제기되는 문제가 있습니다.

Alice가 고개를 끄덕였습니다. 어떤 문제인지 잘 알고 있었습니다.

Malorean Corp.의 ERP는 재고와 생산 작업을 관리했지만 완제품을 만드는 데 사용된 원재료 로트를 제대로 추적할 수 없었습니다.

식품 제조업체에서는 공급업체가 특정 원재료 로트에 결함이 있다고 알리면 그 원재료로 만든 제품을 신속하게 찾아야 했습니다.

현재 Malorean Corp.의 고객들은 자체적으로 이 정보를 다시 구성해야 했습니다.

— 지난주에도 이 문제에 관한 의견을 받았습니다. Hubert가 설명했습니다. 한 고객은 공급업체 경보와 관련된 제품을 찾느라 품질 책임자와 꼬박 하루를 보냈다고 했어요.

— 네. 고객지원팀에서도 비슷한 사례를 여러 건 전달했습니다.

— Lucas에게 맡길 수 있을까요?

Alice는 생각했습니다.

휴가 모듈보다 까다로운 주제였습니다. 화면 몇 개를 만드는 것으로 충분하지 않았습니다. 실제 작업장에서 무슨 일이 일어나는지, 원재료를 어떻게 사용하고 고객이 제품을 어떻게 식별하는지 이해해야 했습니다.

하지만 바로 그 점 때문에 흥미로웠습니다.

— 네, 문제를 맡길 수 있을 것 같습니다.

— 그가 해당 업종을 충분히 이해하고 있을까요?

— 아마 아닐 겁니다.

Hubert가 눈썹을 치켜올렸습니다.

— 안심이 되는군요.

— 저도 세부 사항을 전부 알지는 못해요. 하지만 누구에게 물어야 하는지는 압니다. 제약을 아주 잘 아는 고객들이 있고, 매일 어려움을 접하는 고객지원팀과 구축 담당자도 있습니다. Lucas가 직접 만나면 됩니다.

— 고객과 직접 이야기하게 한다고요?

— 안 될 이유가 있나요?

— 보통은 Alice가 하던 일이잖아요.

— 바로 그래서요.

Hubert는 완전히 납득한 것 같지 않았습니다.

— 필요에 맞지 않는 것을 만들면요?

— 그러지 않도록 돕는 것이 제 역할입니다. 하지만 50쪽짜리 명세서를 써 주지는 않을 겁니다.

— 자신이 만든 해법을 운영 환경에 배포하려 하면요?

— 다른 소프트웨어 변경과 마찬가지로 검토하고 승인해야 합니다. 더 자유롭게 일하게 한다고 모든 위험을 감수할 권리까지 주는 것은 아닙니다.

Hubert는 잠시 말이 없었습니다.

— 제가 무엇을 걱정하는지 알아요? 그가 마침내 물었습니다. Lucas가 실수할 가능성만이 아닙니다. 누가 무엇을 하는지 더 이상 알 수 없게 되는 것이 걱정돼요.

— 저도 걱정됩니다.

— 그래도 해 보고 싶어요?

— 네.

Alice는 말을 덧붙이기 전에 잠시 생각했습니다.

— 다른 선택은 예전과 똑같이 일하면서 개발자에게 코드만 더 빨리 작성하라고 하는 겁니다. 금요일에 Lucas가 보여 준 것이 그것인지는 잘 모르겠어요.

두 사람은 20분 정도 더 이야기했습니다.

Hubert는 진행 중인 프로젝트에 지장을 주고 싶지 않았습니다. Alice는 시작도 하기 전에 회사 전체에 대대적으로 알리는 실험이 되지 않기를 바랐습니다.

그래서 단순하게 시작하기로 했습니다.

Lucas는 로트 추적 문제를 맡기로 했습니다. 해결책을 탐색하고 문제를 이해하는 데 도움을 줄 사람들을 직접 만날 자유를 얻었습니다.

Alice는 제품 측 담당자가 됩니다. 상세한 백로그를 준비해 주지는 않지만 중요한 의사결정에는 계속 참여합니다.

고객에게 출시일을 약속하지 않고 제한된 범위부터 시작합니다.

무엇보다 어떤 일이 벌어지는지 시간을 들여 관찰합니다.

— 그래도 실험의 성공 여부를 어떻게 판단할지 정해야 합니다. Hubert가 말했습니다.

— 맞아요. 개발한 기능의 수만 측정하고 싶지는 않습니다.

— 그러면 무엇을 측정할까요?

— 먼저 문제를 이해하고 쓸모 있는 무언가를 제안할 수 있는지 봐야 합니다. 그다음에는 우리가 모든 것을 대신 정리하지 않아도 어디까지 진전할 수 있는지 살펴봅니다.

— 그리고 우리 시간이 얼마나 드는지도요.

— 그것도요.

Hubert는 시간을 확인했습니다.

— 좋습니다. 해 보죠.

Alice는 완전히 안심하지 못한 채 미소 지었습니다.

— 해 보죠.

— 오늘 Lucas에게 이야기할 건가요?

— 네.

화상회의가 끝났습니다.

Alice는 잠시 화면 앞에 그대로 앉아 있었습니다. 메신저를 열고 Lucas를 찾았습니다.

메시지를 쓰기 시작했습니다.

안녕하세요, *Lucas*. 오늘 잠깐 시간 있어요? 평소와 조금 다른 주제를 하나 제안하고 싶습니다.

보내기 전에 메시지를 다시 읽었습니다.

오랜만에 자신이 해법을 구체적으로 상상하기도 전에 개발자에게 문제를 맡기려 하고 있었습니다.

좋은 생각인지는 아직 알 수 없었습니다.

그리고 이제는 자신이 그보다 먼저 알아야 하는 것이 정말 자신의 역할인지도 확신할 수 없었습니다.

4 – 새로운 조직 모델이 등장하고 있다

AI는 더 많은 실행을 가능하게 합니다. 그렇다면 기업들은 팀과 의사결정 구조를 어떻게 재편하고 있을까요? 그 답은 제각각입니다.

어떤 기업은 개발자 주변의 프로덕트 매니지먼트를 강화합니다. 다른 기업은 개발자의 역할을 넓히고 비기술 직군에 더 많은 수단을 제공하거나, 에이전트의 지원을 받는 소규모 팀을 중심으로 업무를 재편합니다.

역사적으로 굳어진 업무 분담은 더 이상 당연한 것이 아닙니다.

그렇다고 전통적인 모델에 논리가 없었던 것은 아닙니다. 프로덕트 매니지먼트는 고객과 비즈니스 맥락의 상당 부분을 모았고, 디자인은 경험을 구체화했으며, 엔지니어링은 그 의도를 소프트웨어로 구현했습니다. 티켓, 명세, 업무 인계, 회의는 제한된 개발 역량을 중심으로 이 직군들을 조율하는 수단이었습니다.

이런 조직 방식이 갑자기 잘못된 것이 되는 것은 아닙니다.

하지만 실행 역량이 커지면서 여러 기업은 사람과 맥락, 업무를 이전과 다르게 배치하기 시작했습니다.

그러므로 아직은 '올바른 모델'을 찾을 때가 아닙니다.

지금은 새롭게 등장하는 주요 대응 방식을 살펴볼 때입니다.

개발자 주변의 프로덕트 역량 강화하기

가장 직관적인 대응은 프로덕트와 엔지니어링의 비교적 전통적인 구분을 유지하되, 개발자 주변에 배치되는 프로덕트 역량을 늘리는 것입니다.

Gamma는 흥미로운 사례를 보여 줍니다. 이 회사에 따르면 엔지니어 네 명당 프로덕트 매니저가 한 명꼴입니다. 흔히 언급되는 과거의 비율은 대략 열 명당 한 명에 가까웠습니다. [1]

이 모델에서는 엔지니어링 역량이 증가하는 만큼 프로덕트 매니지먼트의 밀도도 높아집니다.

원리는 단순합니다. 더 많은 것을 만들 수 있다면, 그 일을 이해하고 우선순위를 정하며 방향을 제시하는 데에도 더 많은 역량을 투입하는 것입니다.

개발자가 주제의 더 넓은 부분을 맡게 하기

또 다른 전략은 엔지니어의 업무 범위를 넓히는 것입니다.

개발자는 문제가 완성된 해결책으로, 다시 티켓으로 모두 번역될 때까지 반드시 기다리지 않습니다. 필요를 이해하고, 프로토타입을 만들고, 국지적인 판단을 내리고, 결과를 추적하는 일에 더 깊이 참여할 수 있습니다.

Linear는 프로덕트 엔지니어 채용 공고에서 기대하는 책임을 설명합니다. 프로덕트 매니지먼트에 끊임없이 의존하지 않고 문제 정의에 참여하면서, 아이디어부터 출시된 제품까지 하나의 기능을 책임지는 것입니다. [2]

이 모델은 더 많은 맥락과 실행 역량을 한 사람에게 모아 직군 간의 일부 인수인계를 줄이려 합니다.

그렇다고 모든 개발자가 프로덕트 매니저가 된다는 뜻은 아닙니다.

특정 주제에서 엔지니어링 역할의 범위가 넓어지는 모습을 설명하는 것입니다.

직군이 과거의 경계를 넘게 하기

이 변화는 개발자에게만 해당하지 않습니다.

Figma는 역할 변화에 관한 조사에서 응답자의 64%가 자신을 두 가지 이상의 역할과 연결해 인식하며, 3분의 1 이상이 여러 직군에 걸친 책임을 맡고 있다고 답했다고 전합니다. 이 결과를 AI만으로 설명할 수는 없지만, 역할의 경계가 덜 경직되어 가는 현상은 잘 보여 줍니다. [3]

Atlassian은 디자이너가 실제 제품 코드에서 직접 작업하고 엔지니어링 검토에 앞서 스스로 풀 리퀘스트를 여는 워크플로도 소개합니다. [4]

Cursor 역시 프로덕트, 디자인, 엔지니어링이 구축 작업에 더 가까이 다가가는 조직을 설명합니다. [5]

따라서 핵심은 '모두가 모든 일을 한다'기보다 더 많은 사람이 자신의 역사적인 업무 범위를 벗어나 개입한다는 데 있습니다.

이러한 경계의 유연성은 별개의 질문을 낳습니다. 여러 직군이 같은 영역에서 행동할 수 있다면 어떻게 책임을 명확하게 유지할 수 있을까요?

이 문제는 뒤에서 다시 다루겠습니다.

책임 있는 소규모 팀에 더 많은 업무 맡기기

또 다른 대응 방식은 더 적은 인원에게 더 넓은 범위를 맡기는 것입니다.

AMPECO는 엔지니어 한 명과 프로덕트 오너 한 명이 의도를 정렬하는 'alignment sync'를 진행한 뒤, 에이전트가 분석과 개발, 테스트의 상당 부분을 맡는 방식을 설명합니다. [6]

Taito는 팀이 계획을 승인하면 한두 명이 에이전트와 함께 구축 작업을 이끄는 운영 방식을 소개합니다. [7]

이런 모델에서는 소규모의 인간 단위가 이전보다 더 많은 일을 움직일 수 있습니다.

여기서 관찰되는 조직 변화는 직군 사이의 인수인계를 줄일 가능성과, 한 사람이나 소규모 집단을 중심으로 주제가 집중되는 현상입니다.

다만 이 사람들이 무엇을 스스로 결정할 수 있는지, 언제 해당 분야의 책임자에게 도움을 요청해야 하는지는 여전히 정의해야 합니다.

검증의 일부 자동화하기

생산 속도가 빨라지면 검증 체계에도 압력이 가해집니다.

일부 기업은 더 많은 검사를 자동화하거나 변경 사항의 위험도에 따라 처리 방식을 달리하는 것으로 대응합니다.

Meta는 변경 사항이 장애를 일으킬 위험을 추정하기 위해 Diff Risk Score를 사용합니다. [8]

Cloudflare는 병합 전에 자동으로 검사할 수 있는 엔지니어링 표준을 정립했습니다. [9]

Taito 역시 결정론적 규칙, 검토 에이전트, 위험 트리거를 결합합니다. [7]

이 단계에서 중요한 사실은 단순합니다. 검증이 더 이상 반드시 모든 경우에 똑같은 방식으로 이루어지지는 않습니다.

자동화와 인간의 책임, 에스컬레이션 기준을 결합하는 방법은 뒤에서 더 자세히 살펴보겠습니다.

모든 단계를 지시하는 대신 맥락 제공하기

어떤 기업들은 한 걸음 더 나아가 상세한 티켓이 업무 조율의 주된 수단이어야 한다는 생각 자체에 의문을 제기합니다.

Linear는 프로덕트 매니저가 문제의 범위를 정하고 시스템에 기록한 뒤 엔지니어링이 실행하기를 기다리는 모델을 명시적으로 비판합니다. 이 회사는 고객 피드백, 아이디어, 전략 방향, 의사결정, 코드에 인간과 에이전트가 모두 접근할 수 있어야 한다고 강조합니다. [10]

Cursor 역시 더 많은 국지적 주도권을 가능하게 하려면 목표가 충분히 명확해야 한다고 강조합니다. [5]

그렇다고 모든 조율이 사라지는 것은 아닙니다.

조율의 일부가 상세한 지시에서 더 풍부하고 더 직접적으로 접근할 수 있는 맥락으로 옮겨 가는 것입니다.

그 맥락에 무엇이 들어가야 하는지, 어떻게 유지해야 하는지, 자율성의 한계를 어떻게 정해야 하는지는 별도의 장에서 다루겠습니다.

필요를 아는 사람이 직접 만들 수 있게 하기

어떤 기업들은 필요를 가장 잘 아는 팀이 개발자의 대기열을 반드시 거치지 않고도 작은 도구를 직접 만들 수 있게 합니다.

Shopify는 더 많은 직원이 웹사이트와 소규모 도구를 빠르게 게시할 수 있는 내부 플랫폼 Quick을 만들었습니다. [11]

Vercel은 비기술 직군의 직원이 주력 제품의 우선순위에서 개발자를 떼어 놓지 않고도 안전한 환경에서 내부 도구를 구축하는 모델을 설명합니다. [12]

Atlassian 역시 기술 검토 전에 디자이너가 코드에 직접 참여하는 사례를 보여 줍니다. [4]

이 경우 AI는 개발자를 더 빠르게 만드는 데 그치지 않습니다.

때로는 이미 필요의 맥락을 가진 사람에게 실행을 옮길 수 있게 합니다.

AI보다 오래된 원칙

2018년 Marty Cagan은 이미 자신이 *empowered product teams*라고 부른 모델을 옹호했습니다. 경영진이 출시할 기능 목록이 아니라 해결할 문제를 맡기는 다직군 팀입니다. 그는 이 팀들이 프로덕트, 디자인, 엔지니어링 역량을 활용해 해결책을 찾고, 생산량이 아니라 결과에 책임져야 한다고 주장했습니다. [13]

그는 Chris Jones와 함께 2020년 출간한 *EMPOWERED*에서 이 접근법을 더 발전시켰습니다. 이 모델은 직군이 서로 구분되지 않게 되거나 관리자가 사라지는 미래를 예고한 것이 아닙니다. 오히려 공유된 비전과 탄탄한 역량, 팀의 성장을 돕는 책임자를 전제로 했습니다. [14]

그러므로 이 아이디어는 에이전트보다 먼저 존재했습니다. AI가 바꿀 가능성이 있는 것은 자율적인 팀이 탐색하기로 한 해결책 일부를 더 빠르고 직접적으로 실행할 수 있는 역량입니다. 프로덕트 자율성은 더 오래된 지향점이며, AI는 그 수단과 제약을 바꿉니다.

아직 하나로 수렴하지 않은 선택들

이 대응 방식들은 하나의 변화 과정에 속한 단계가 아닙니다. 프로덕트 역량을 강화하는 것과 개발자에게 더 많은 프로덕트 책임을 맡기는 것은 서로 모순처럼 보일 수도 있습니다. 그러나 서로 다른 기업에서 각각 효과를 낼 수 있고, 심지어 같은 조직 안에서 공존할 수도 있습니다.

공통점은 다른 곳에 있습니다. 실행이 더 쉬워지면 맥락과 의사결정, 책임을 배분하는 방식이 다시 선택의 문제가 됩니다.

이제 공표된 방침과 조직도를 넘어, 이러한 이동이 당사자들의 일상에 무엇을 가져오는지 살펴봐야 합니다.

곧 Malorean Corp.로 돌아가 이런 모델이 프랑스의 한 중소기업에서 일하는 사람들의 삶을 어떻게 바꾸는지 관찰하겠습니다. 이어서 이러한 자율성에 필요한 책임과 규칙을 살펴보겠습니다.

참고문헌

[1] The bottleneck keeps shifting: What AI is changing about how we build – Atlassian / Gamma

- URL: https://www.atlassian.com/blog/how-we-build/the-bottleneck-keeps-shifting-what-ai-is-changing-about-how-we-build
- 엔지니어링 생산 역량이 커질 때 Gamma가 프로덕트 매니지먼트의 밀도를 높인 사례를 설명합니다.

[2] Product Engineer – Linear Careers

- URL: https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- 문제 정의부터 출시에 이르기까지 프로덕트 엔지니어에게 기대되는 책임을 제시한 채용 공고입니다.

[3] Are Roles and Responsibilities a Thing of the Past? – Figma

- URL: https://www.figma.com/blog/2025-shifting-roles-report/
- 역할의 경계가 점점 유연해지고 책임이 전통적인 직군의 경계를 넘어 확장되는 현상을 다룹니다.

[4] Designers' workflow for shipping code – Atlassian

- URL: https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code

- 디자이너가 엔지니어링 검토 전에 제품 코드에 직접 참여하는 방식을 설명합니다.

[5] How Cursor Is Navigating the New Shape of Product Craft – Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- 프로덕트, 디자인, 엔지니어링이 구축 작업에 더 가까이 참여하는 운영 방식을 보여 줍니다.

[6] How we built an AI-native engineering system – AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- 소수의 사람이 의도를 정한 뒤 실행의 상당 부분을 에이전트에게 맡기는 조직을 설명합니다.

[7] Inside an AI-native engineering organization – Swarmia / Taito.ai

- URL: https://www.swarmia.com/blog/inside-an-ai-native-engineering-organization/
- 한두 명을 중심으로 실행의 상당 부분을 집중시키고 소규모 팀이 에이전트를 이끄는 방식을 설명합니다.

[8] Diff Risk Score: AI-driven risk-aware software development – Meta

- URL: https://engineering.fb.com/2025/08/06/developer-tools/diff-risk-score-drs-ai-risk-aware-software-development-meta/
- 변경 사항의 예상 중요도에 따라 검증 워크플로를 달리하는 데 사용하는 위험 점수를 소개합니다.

[9] How Cloudflare enforces engineering standards using AI – Cloudflare

- URL: https://blog.cloudflare.com/engineering-standards-enforcement/

- 명문화된 엔지니어링 표준이 개발 워크플로의 자동화된 검사로 바뀌는 과정을 보여 줍니다.

[10] Issue tracking is dead – Linear

- URL: https://linear.app/next
- 고객 피드백과 전략, 의사결정, 코드가 고립된 티켓을 넘어 더 많은 맥락을 공유하는 모델을 옹호합니다.

[11] Quick: An internal hosting platform for the AI era – Shopify

- URL: https://shopify.engineering/quick
- 더 많은 직원이 직접 만들고 게시할 수 있게 하는 내부 플랫폼을 소개합니다.

[12] Stopping the slow death of internal tools – Vercel

- URL: https://vercel.com/blog/stopping-the-slow-death-of-internal-tools
- 비즈니스 사용자가 엔지니어링을 주력 로드맵에서 떼어 놓지 않고도 내부 도구를 만들 수 있는 방식을 보여 줍니다.

[13] Empowered Product Teams – Marty Cagan, Silicon Valley Product Group (2018)

- URL: https://www.svpg.com/empowered-product-teams/
- 문제 해결의 책임을 맡은 팀과 기능 출시에만 머무는 팀을 구분한 기초적인 글입니다.

[14] EMPOWERED: Ordinary People, Extraordinary Products – Marty Cagan et Chris Jones (2020)

- URL: https://www.svpg.com/books/empowered-ordinary-people-extraordinary-products/
- 자율적인 프로덕트 팀과 이들에게 맡기는 목표, 전략적 맥락, 팀의 성장을 돕는 책임자의 역할을 다룬 책을 소개합니다.

5 – Malorean Corp. – 할 수 있는 일과 해야 하는 일

화요일 아침, Lucas는 Alice가 뜻밖이라고 느낀 질문으로 이야기를 시작했습니다.

— 고객과 이야기해 봐도 될까요?

전날 그는 로트 추적 문제에 관해 지원팀이 보관하고 있던 몇 건의 보고서를 받았습니다. 메시지에는 제조 번호, 원자재, Excel 파일, 공급업체 경보에 관한 내용이 담겨 있었습니다. 그는 AI의 도움을 받아 저녁 내내 자료를 정리했습니다.

— 물론이지. 왜? Alice가 대답했습니다.

— 정확히 무엇이 부족한지 이해가 안 돼서요.

Alice는 미소를 지었습니다. 일주일 전이었다면 아마 자신이 먼저 이 미팅을 진행한 뒤 결론을 개발팀에 전달했을 것입니다. 하지만 이제는 Lucas에게 이 주제를 맡기겠다고 약속한 상태였습니다.

— 이 고객을 담당하는 지원팀 직원에게 소개해 줄게. 그분이 미팅을 잡아줄 수 있을 거야.

그렇게 수요일이 되자 Lucas는 '로트 관리'라는 말보다 훨씬 복잡한 상황을 마주했습니다.

고객사의 품질 책임자는 공급업체에서 받은 서신을 보여 주었습니다. 어떤 원자재를 제외해야 한다는 내용이었습니다. 공급업체는 해당 로트 번호를 알고 있었고, 제조업체는 어떤 주문이 입고되었는지 알고 있었습니다. 하지만 정확히 어떤 완제품이 영향을 받았는지 알아내려면 직원들이 여러 문서를 대조해야 했습니다.

— 지난번에는 네 명이 하루 중 상당 시간을 파일 확인에 썼습니다. 품질 책임자가 설명했습니다. 그래도 확신이 서지 않아 작업장에 다시 가서 확인해야 했고요.

Lucas는 제조 시점에 정보가 어떻게 기록되는지 보여 달라고 요청했습니다. 고객은 ERP 화면 하나를 보여 준 뒤 스프레드시트도 열었습니다.

— 그런데 이 번호를 Excel에도 기록하는 이유가 뭔가요?

— 귀사의 소프트웨어로는 저희가 찾고 싶은 것을 찾을 수 없기 때문입니다.

공격적인 대답은 아니었습니다. 그래서 오히려 더 듣기 어려웠습니다.

통화가 끝났을 때 Lucas에게는 세 페이지 분량의 메모가 있었지만, 확신은 시작할 때보다 줄어 있었습니다.

— 그래서 어땠어? Lucas가 다시 전화하자 Alice가 물었습니다.

— 처음에는 로트 번호를 등록하는 화면을 생각하고 있었어요.

— 지금은?

— 지금은 그들이 실제로 사용한 원자재를 어떻게 기록하는지부터 이해해야 할 것 같아요. 그렇지 않으면 멋진 화면은 생겨도 여전히 그들의 질문에는 답하지 못할 테니까요.

Alice는 고개를 끄덕였습니다. 월요일에 자신이 직접 말해 줄 수도 있었습니다. 하지만 Lucas가 스스로 깨달은 것이 싫지는 않았습니다.

다음 금요일, Lucas는 데모 데이터베이스에서 만든 첫 번째 버전을 보여 주었습니다.

원자재 로트를 제조 작업에 연결하고, 그 원자재로 만든 완제품을 역으로 찾을 수 있었습니다. 그는 경보 대상이 된 로트를 관련 제품까지 추적하는 간단한 예시도 추가했습니다.

Alice는 몇 차례 클릭해 보았습니다.

— 이번 주에 이걸 다 만든 거야?

— 네. 아직 틀렸거나 빠진 것이 많지만, 고객에게 보여 줄 무언가는 생겼어요.

— 반대 방향으로도 추적할 수 있게 하면 어떨까?

— 이미 돼요.

그는 다른 화면을 열었습니다.

Alice는 웃음을 터뜨렸습니다.

— 역시 그렇겠지.

휴가 모듈 시연 이후 Alice는 프로젝트의 규모를 상상하는 방식이 달라졌습니다. 예전 같으면 다음 분기로 미뤘을 일이 이제는 손에 닿을 듯했습니다.

— 공급업체별로 검색할 수도 있겠네. 관련 주문에 바로 경보를 표시할 수도 있고. 리콜 문서까지 준비하면 어떨까?

Lucas는 바로 대답하지 않았습니다.

— 아이디어는 적어 둘 수 있어요. 그가 마침내 말했습니다. 하지만 먼저 지금 만든 걸 가지고 고객을 다시 만나고 싶어요.

— 더 완성된 버전을 보여 주려고 2주나 기다릴 수는 없잖아?

— 시연이 걱정되는 게 아니에요.

— 그럼 뭐가 걱정인데?

이번에 Lucas가 공유한 화면은 보기에 그다지 좋지 않았습니다. 오래된 테이블과 의미를 알 수 없는 이름의 필드, 제조와 재고를 연결하는 여러 관계가 보였습니다.

8

— 화면은 원하는 만큼 만들 수 있어요. 문제는 우리 ERP가 로트의 전체 이력을 보관하도록 설계된 적이 없다는 거예요. 몇 년 전의 재고 이동 기

록도 있고, 때로는 수작업으로 수정된 것도 있어요. 어떤 정보는 아예 존재하지 않고요.

— AI로 이 모든 걸 정리할 수 있지 않을까?

— 마이그레이션을 작성하는 데 도움은 받을 수 있어요. 하지만 한 번도 기록하지 않은 정보를 만들어 낼 수는 없어요.

Alice는 말이 없었습니다.

— 제대로 하려면 데이터 모델 일부를 바꾸고 다른 모듈에 어떤 영향이 생기는지도 확인해야 해요. Lucas가 말을 이었습니다. 2주를 아끼려고 고객사의 재고를 틀리게 만들 수는 없잖아요.

— 소프트웨어를 망가뜨리라고 한 건 아니야.

— 알아요. 하지만 금요일 이후로 제가 하나를 보여 드릴 때마다 세 가지를 더 요청하고 계세요.

공격적인 말투는 아니었습니다. 그래도 Alice의 마음을 찔렀습니다.

— 맞아. Alice가 인정했습니다.

— 저도 빨리 가고 싶어요. 잘 아시잖아요. 하지만 한 시간 만에 코드를 생성할 수 있다고 해서 15년 동안 운영해 온 ERP가 단순해지는 건 아니에요.

그는 잠시 망설이다가 덧붙였습니다.

— Thomas가 휴가 모듈 코드 일부를 살펴봤어요. 우리가 잘못 처리한 사례 두 가지를 찾아냈고, 그의 말이 맞았어요. 수정은 했지만 한 가지를 다시 깨달았어요. 훨씬 빠르게 만들 수 있게 되어도 함정을 아는 사람은 여전히 필요하다는 거예요.

Alice는 시연이 있던 금요일, Thomas가 그저 코드를 보여 달라고 했던 일을 떠올렸습니다.

— 그럼 어떻게 하자는 거야?

— 가상 데이터로 시연 작업은 계속해요. 우리가 필요를 제대로 이해했는지 확인하고요. 하지만 실제 재고를 건드리려면 기존 시스템을 이해하고 필요한 검사를 하는 데 시간을 들이고 싶어요.

Alice는 다른 화면에 아직 열려 있는 첫 번째 버전을 바라보았습니다.

지난주 Alice는 Hubert 앞에서 Lucas의 자율성을 옹호했습니다. 그 자율성에 자신에게 '안 된다'고 말할 권리까지 포함된다는 사실은 그다지 반갑지 않았습니다.

— 좋아. Alice가 마침내 말했습니다. 당분간 실제 데이터는 건드리지 말자.

다음 월요일, Hubert는 이제 익숙해진 질문으로 주간 회의를 시작했습니다.

— 그래서, 우리 실험은 어떻게 되고 있나요?

— 진행 중이에요. 어떤 부분은 예상보다 빠르고, 다른 부분은 더 느려요. Alice가 대답했습니다.

— 무슨 뜻이죠?

Alice는 Lucas의 시연을 보여 준 뒤 금요일에 나눈 대화를 요약했습니다.

Hubert는 기술적인 우려에 오래 머물렀습니다.

— 조심하는 게 맞아요. 그는 말했습니다. 추적 문제 하나를 해결하다가 모든 고객의 재고 문제를 만들고 싶지는 않으니까요.

— 네. 제가 조금 몰아붙였던 것 같아요.

— Alice가요?

— 지난 2주 동안 모든 것이 가능해진 듯한 기분이 들었어요. 아이디어가 하나 떠오를 때마다 당장 시험해 볼 수 있다고 생각하게 돼요.

Hubert는 미소를 지었습니다.

— 첫 번째 시연을 본 뒤 저도 그랬어요.

— 문제는 화면에 무언가가 나타나는 속도와 신뢰할 수 있는 무언가를 얻는 속도를 혼동하기 쉽다는 거예요.

— 적어도 Lucas는 언제 속도를 늦춰야 하는지 아는군요.

— 네. 그 점에서는 제가 Lucas를 믿어야 해요.

이틀 뒤 Julie Morel이 Alice에게 5분만 시간을 내 달라고 했습니다.

Alice는 이런 표현에 익숙했습니다. Julie에게 5분은 쉽게 30분이 될 수 있었습니다.

— 지난주에 꽤 큰 잠재 고객을 만났어요. Julie가 말을 꺼냈습니다. 공장이 여러 곳인데, 일하는 방식에 맞춰 조정할 수 있는 ERP를 찾고 있어요. 제가 준비한 걸 보세요.

Julie는 아주 깔끔한 목업 여러 장을 공유했습니다. 주문 종류에 따라 다른 승인 단계, 사업장별 규칙, 사용자에 따라 달라지는 화면이 보였습니다.

— 이걸 네가 만들었어? Alice가 물었습니다.

— ChatGPT와 함께요. 미팅 메모를 넣고 사용자 흐름을 제안해 달라고 한 다음 함께 수정했어요. 저녁 한 번을 썼죠.

Alice는 이 이미지가 어떤 가치를 주는지 즉시 알아보았습니다. Julie는 몇 시간 만에 영업 보고서 안에 갇혀 있던 필요를 구체적인 형태로 만들었습니다.

— 고객도 이 목업을 봤어?

— 아직이요. 먼저 보여 드리고 싶었어요. 하지만 이런 걸 할 수 있다면 이야기가 달라질 수 있다고 분명히 말했어요.

Alice는 화면들을 넘겨 보았습니다.

— 아이디어가 흥미롭네. Lucas와 직접 이야기해 봐.

Julie는 놀란 눈으로 바라보았습니다.

— 직접요? Alice가 주제를 정리하실 줄 알았는데요.

— 왜? 네가 고객을 알고, 직접 이야기했고, 목업까지 준비했잖아. 고객이 무엇을 찾는지 나보다 네가 더 잘 설명할 수 있어.

— Lucas는 가능한지 살펴볼 수 있고요?

— 그래. 하지만 Julie, 조심해야 해. 가능한지 살펴본다고 해서 개발을 약속하는 건 아니야.

— 물론이죠.

Alice는 Julie가 말하는 '물론'이 어떤 것인지 알고 있었습니다. 거짓말은 아니었습니다. 가능한 계약을 눈앞에 둔 영업 담당자의 열정이었습니다.

다음 날, 세 사람은 함께 모였습니다. Julie는 목업을 차례로 보여 주며 잠재 고객의 요청을 설명하고 질문에 답했습니다.

Lucas는 사업장마다 규칙이 어떻게 달라지는지에 곧바로 관심을 보였습니다.

— ERP에 이미 이 메커니즘의 일부가 있어요. 그가 말했습니다. 정확히 이런 일을 위한 것은 아니지만 존재하긴 해요. 시험해 볼 무언가를 준비할 수 있을 것 같아요.

— 얼마나 걸려요? Julie가 물었습니다.

— 시연용이라면 그렇게 오래 걸리지는 않을 거예요.

Julie는 Alice를 바라보았습니다. 미소는 이미 계약을 따낸 사람의 표정과 거의 같았습니다.

— 다음 주에 이걸 보여 주면 정말 큰 차이를 만들 수 있어요.

Alice는 생각하기도 전에 앞으로 나아가고 싶은 자신의 마음이 먼저 대답하려는 것을 느꼈습니다. 추적 프로젝트와 밀린 요청들, 이 잠재 고객만의 특별한 규칙, 그리고 그 기능을 전혀 쓸 일이 없을지도 모르는 다른 모든 고객을 떠올렸습니다.

— 안 돼. Alice가 말했습니다.

두 사람 모두 그녀를 바라보았습니다.

— 뭐가 안 된다는 거예요? Julie가 물었습니다.

— 이 계약을 따내기 위해 우리 제품에 이 기능을 개발하기 시작하는 것.

— 하지만 방금 흥미롭다고 하셨잖아요.

— 흥미로워. 하지만 이 고객에게 모든 고객이 써야 할 기능이 필요한 건지, 그들의 공장에만 맞는 아주 특별한 운영 방식이 필요한 건지 아직 몰라.

— 계약을 놓칠 수도 있어요.

— 그럴 수도 있지.

Julie는 분명 이런 대답을 예상하지 못했습니다.

— 이제야 이런 요청에 빠르게 대응할 수 있는 수단이 생겼잖아요. Julie가 주장했습니다. 쓰지 않는 건 아까워요.

— 쓰지 않겠다는 게 아니야. 우리가 무엇을 제안하는지 이해하고 싶다는 거야. 특별 요청을 전부 ERP 핵심에 추가하면 결국 발전시킬 수 없는 소프트웨어가 될 거야.

— 하지만 Lucas는 할 수 있다고 했어요.

— Lucas는 시연을 준비할 수 있다고 했어. 같은 말이 아니야.

잠시 침묵이 흘렀습니다. Julie는 목업을 한 번 더 바라보았습니다.

— 그럼 고객에게 뭐라고 말하죠?

— 문제를 이해했고 한 가지 가능성을 보여 주고 싶다고 해. 이 기능을 제품에 추가하기로 결정했다고 말하지는 말고.

Julie는 고개를 끄덕였습니다. 실망한 듯했지만 낙담하지는 않았습니다.

— 알겠어요. 공장마다 무엇이 다른지 더 자세히 설명해 달라고 미팅을 잡을 수 있어요.

— 좋아. 목업도 그대로 두고. 올바른 질문을 하는 데 도움이 될 거야.

Julie가 통화를 마친 뒤 Alice는 잠시 Lucas와 둘만 남았습니다.

— 정말 이걸 빨리 만들 수 있었어?

— 첫 번째 버전이라면요.

— 만들고 싶었고?

— 솔직히요? 네. 재미있어 보였어요.

Alice는 웃었습니다.

— 나도 그러자고 하고 싶었어.

— 그런데 왜 안 된다고 하셨어요?

— 우리 둘 다 할 수 있는 일만 이야기하고 있었지, 해야 하는 일은 거의 이야기하지 않고 있다는 걸 깨달았거든.

다음 주, Lucas는 메시지와 함께 Alice에게 링크 하나를 보냈습니다.

어제 뭔가 하나 만들어 봤어요. 한번 보실래요?

Alice가 페이지를 열자 익숙한 ERP 인터페이스 옆에 몇 가지 규칙을 바꿀 수 있는 작은 패널이 보였습니다. 사업장을 선택하고, 승인 기준을 바꾸고, 단계를 표시하거나 숨긴 뒤 시연을 다시 실행할 수 있었습니다.

Alice는 Lucas에게 전화했습니다.

— 이게 뭐야?

— 규칙을 바꿔 가며 시연을 준비할 수 있는 작은 도구예요. 기존 설정 메커니즘을 재사용했어요. 테스트 인스턴스에서만 돌아가고요.

— 얼마나 걸렸어?

— 두 시간 정도요. Julie의 목업을 다시 생각하다가 아이디어가 떠올랐어요.

— 그런데 이건 네가 맡은 주제도 아니잖아!

— 네. 변형을 시험할 때마다 코드를 수정하지 않을 방법이 있는지 보고 싶었어요.

Alice는 매개변수를 바꾸기 시작했습니다. 화면이 그에 맞춰 반응했습니다. 다른 설정을 시험한 뒤 세 번째 설정도 적용해 보았습니다.

— 잠깐만. 내가 직접 여러 시연 버전을 준비할 수 있다는 뜻이야?

— 노출해 둔 규칙에 대해서는요. 무엇이든 할 수 있는 건 아니에요. 특히 고객 환경에서는 절대 안 되고요.

— 하지만 Julie에게 서로 다른 흐름 두 가지를 보여 주고 어느 쪽이 잠재 고객에게 맞는지 물어볼 수 있겠네.

— 정확해요.

Alice는 잠시 페이지를 바라보았습니다. 첫 번째 휴가 모듈은 Lucas를 더 빠르게 만들었습니다. 이 도구는 어제까지 Alice에게 없던 가능성을 주었습니다.

개발자에게 자신만을 위한 구현을 부탁하지 않고도 아이디어를 시험해 볼 수 있게 된 것입니다.

— 이걸로 무엇을 할 수 있을지 상상이 돼?

Lucas는 미소를 지었습니다.

— 그럴 것 같아요. 하지만 완전한 제품으로 만들어 달라고 하시기 전에, 두 시간 만에 만든 임시 도구라는 점을 말씀드릴게요.

— 약속할게.

통화를 마친 Alice는 다시 매개변수를 바꾸며 시험하기 시작했습니다.

다음 월요일, Hubert는 Alice가 평소의 진행 상황 문서를 열지 않은 것을 알아차렸습니다.

— 오늘은 준비한 게 없나요?

— 있어요. 하지만 보여 드리고 싶은 게 있어요.

Alice는 Lucas의 도구를 소개하고, Hubert가 보는 앞에서 같은 시연의 변형 두 가지를 만들었습니다.

Hubert는 금세 이해했습니다.

— 이 변경을 개발자들에게 부탁하지 않은 건가요?

— 네. Lucas가 작은 도구를 만들었어요. 몇 가지 규칙을 저 혼자 시험할 수 있어요.

— 영업팀도 쓰게 할 수 있을까요?

— 나중에는 가능할지도 몰라요. 지금은 충분히 견고하지 않아요. 하지만 Julie는 잠재 고객 사례를 놓고 저와 함께 작업할 수 있어요.

Hubert는 흥미로운 표정으로 화면을 바라보았습니다.

— 로트 프로젝트는요?

— Lucas가 진행하고 있어요. 첫 번째 버전을 들고 고객을 다시 만났고, 이제 무엇을 해야 하는지 더 잘 이해하고 있어요. 하지만 데이터 처리 방식을 안전하게 만들기 전에는 과거 데이터를 건드리지 않겠다고 해요.

— 합리적으로 들리네요.

— 네. 금요일에는 짜증이 났지만, 오늘은 Lucas가 고집을 부려서 다행이라고 생각해요.

Alice는 손가락 사이로 펜을 굴렸습니다.

— 이상한 일이 벌어지고 있어요, Hubert. Lucas가 고객과 이야기하고 해결책에 관한 결정을 내려요. Julie는 거의 바로 쓸 수 있는 목업을 가져오고요. 이제 저는 개발 과정을 거치지 않고도 ERP의 몇 가지 변형을 시험할 수 있어요.

— 걱정되나요?

— 때로는요. 모두가 다른 사람 일의 일부를 하기 시작하면 우리 직업은 어떻게 되는지 궁금해져요.

— 답을 찾았나요?

— 아직은 아니에요.

Alice는 잠시 생각했습니다.

— 모든 아이디어가 반드시 저를 거쳐야 하는 사람이 되고 싶지는 않다는 것만 알아요. Julie가 자신의 목업을 제가 티켓으로 바꿀 때까지 기다릴 이유는 없었어요. Lucas에게도 화면 하나하나를 어떻게 만들지 제가 설명해 줄 필요는 없고요.

— 그래도 잠재 고객의 요청을 개발하는 건 거절했죠.

— 네. 우리 제품에 관한 결정이었으니까요. 제가 틀릴 수도 있고, 생각을 바꿀 수도 있어요. 하지만 그 선택은 제가 책임져야 했어요.

— Lucas도 재고 문제에서 지름길을 택하는 걸 거절했고요.

— 무엇을 망가뜨릴 위험이 있는지 알고 있으니까요. 그 부분에서는 제가 들어야 할 판단력을 갖고 있어요. 제가 아무리 급하더라도요.

Hubert는 작은 창 속 Alice의 얼굴을 잠시 바라보았습니다.

— 그렇다면 사람들에게 더 많은 일을 맡길 수는 있지만, 모든 결정의 책임이 그들 사이를 떠돌게 둘 수는 없다는 말이군요.

— 저도 그렇게 생각하기 시작했어요.

Alice는 망설이다가 덧붙였습니다.

— 예전에는 많은 주제가 저를 거쳐 가기 때문에 제 일이 중요하다고 느낄 때가 있었어요. 이제는 마지막에 도달하는 결과가 더 중요하지 않은지 궁금해요. 올바른 문제를 해결하고 있는가? 신뢰할 수 있는 것을 만들고 있는가? 할 가치가 있는 일인가?

Hubert는 고개를 끄덕였습니다.

— 이 질문들에 누가 답해야 하는지 아무도 모른다면 문제가 생기겠죠.

— 네.

두 사람은 몇 초 동안 말없이 있었습니다. 어느 쪽도 이 대화를 새로운 조직 개편 발표로 바꾸고 싶지 않았습니다.

그들의 실험은 이제 막 시작되었습니다. 실험은 더 많은 가능성을 가져왔지만, 내려야 할 결정도 더 많아졌습니다.

Alice에게 알림이 왔습니다. Julie가 보낸 메시지였습니다. 잠재 고객이 서로 다른 시연 버전을 보겠다고 했고, 언제부터 자기 팀에서 사용할 수 있는지도 알고 싶어 했습니다.

Alice는 바로 답하지 않았습니다.

이제 소프트웨어는 훨씬 더 빠르게 답을 만들어 낼 수 있었습니다.

하지만 Malorean Corp.가 어떤 답을 약속하고 책임질 준비가 되어 있는지는 여전히 누군가가 결정해야 했습니다.

6 – 역할의 경계는 흐려져도 책임은 남는다

오랫동안 프로덕트, 디자인, 엔지니어링 사이의 경계는 단순한 기준점 역할을 했습니다.

프로덕트는 문제를 이해하고 무엇을 만들지 결정했습니다. 디자인은 경험을 설계했습니다. 엔지니어링은 그 의도를 소프트웨어로 구현했습니다. 현실에서 이 구분이 언제나 이처럼 깔끔했던 것은 아니지만, 그래도 팀과 채용, 프로세스, 책임의 구조를 만들었습니다.

이러한 업무 분담은 AI 이전부터 이미 도전받고 있었습니다. 1.4장에서 설명한 자율적인 프로덕트 팀은 프로덕트, 디자인, 엔지니어링이 하나의 공통된 문제를 놓고 함께 일하게 했습니다. 이제 AI는 또 다른 경계, 즉 과거에는 특정 직군에만 속했던 일을 각자가 더 많이 수행할 수 있게 되면서 실행의 경계를 넘는 움직임을 가속합니다.

조직 모델을 살펴본 앞 장에서는 특히 Figma의 연구를 통해 이러한 변화를 확인했습니다. [1] 저는 Malorean에서도 개발자는 문제를 이해하는 쪽으로, 프로덕트 매니저는 문제를 구현하는 쪽으로 서로 다가가는 움직임을 설명한 바 있습니다. [2]

이를 보고 역할 자체가 사라질 것이라고 결론 내리고 싶을 수도 있습니다. 하지만 저는 이 변화가 각자의 책임을 다시 정의하도록 요구한다고 생각합니다. 역할의 경계가 유연해질수록 책임은 더 명시적이어야 합니다.

행동할 역량은 더 이상 한 직군에만 속하지 않는다

AI가 가장 먼저 재분배하는 것은 반드시 의사결정 권한이 아닙니다. 행동할 역량입니다.

Atlassian이 소개한 것처럼 디자이너가 직접 코드를 수정하는 사례를 생각해 봅시다. 기술팀은 여전히 품질과 운영 환경 배포 조건을 검토합니다. [3] 이 단계는 과거의 업무 구분으로 돌아간다는 뜻이 아닙니다. 코드에

개입할 역량과 그 신뢰성을 책임지는 일은 서로 같지 않다는 사실을 보여줍니다.

Linear의 프로덕트 엔지니어 채용 공고에서도 같은 구분을 볼 수 있습니다. 이 회사는 최초의 문제부터 출시된 결과까지 하나의 주제를 맡을 역량을 기대합니다. [4] 그렇다고 그 직무에 모든 프로덕트 전략 결정을 내릴 권한이 자동으로 생기는 것은 아닙니다.

따라서 일부 업무 인계를 줄이면서도, 그 인계가 필요했던 이유인 전문성까지 없애지는 않을 수 있습니다. 이제 중요한 것은 몇 개의 직군이 참여하는지 세는 일이 아니라, 여러 직군이 같은 영역에 개입할 때 누가 무엇을 책임지는지 아는 것입니다.

기여할 수 있다는 것이 그 영역을 책임진다는 뜻은 아니다

조직이 자주 혼동하는 두 가지를 구분해야 합니다. 행동할 권한과 책임입니다.

영업 담당자는 내부 도구를 만들 수 있습니다. 그렇다고 플랫폼 보안을 책임지는 사람이 되는 것은 아닙니다.

프로덕트 매니저는 인터페이스를 수정할 수 있습니다. 그렇다고 디자인 시스템의 일관성을 책임지게 되는 것은 아닙니다.

개발자는 국지적인 제품 동작을 결정할 수 있습니다. 그렇다고 전체 전략과 포지셔닝, 로드맵을 자동으로 책임지는 것은 아닙니다.

많은 팀에서 이미 존재하던 구분이지만, AI는 이를 명문화하도록 만듭니다.

고도로 전문화된 업무 방식에서는 직군과 책임이 흔히 겹쳤습니다. 행동할 수 있는 사람이 곧 그 영역의 담당자이기도 했습니다. 행동할 역량이 널리 퍼지는 순간 이 중첩은 사라집니다.

그러면 자율성을 책임의 희석과 혼동할 위험이 생깁니다.

저는 품질에 관해서도 이와 비슷한 원칙을 주장한 적이 있습니다. 개발자가 무언가를 출시한 뒤 결함은 QA 팀이나 버그 당번, 혹은 다음으로 시간

이 나는 개발자의 일이라고 여겨서는 안 됩니다. 출시한 것의 기술 품질은 여전히 식별 가능한 누군가의 책임입니다. [5]

역할의 경계가 유연해져도 이 원칙은 달라지지 않습니다. 오히려 더 중요해집니다.

보증인이라는 개념

저는 이 문제에 답하기 위해 보증인이라는 개념을 사용합니다.

중요한 영역마다 한 사람이나 한 팀이 그 일관성과 품질을 계속 책임집니다.

보증인이 반드시 그 영역의 모든 작업을 수행하는 사람일 필요는 없습니다.

사소한 결정 하나하나를 승인해야 하는 사람도 아닙니다.

그 영역에 영향을 미치는 중요한 선택이 있을 때, 한계를 넘어섰을 때, 혹은 무언가 잘못되었을 때 조직이 찾는 사람입니다.

단순한 조직이라면 적어도 세 가지 보증인을 명시할 수 있습니다.

- 프로덕트 보증인: 프로덕트의 일관성, 구조적 선택, 타깃, 가치 제안, 중대한 조정, 전략과의 관계.
- 디자인 보증인: 디자인 시스템, 경험의 일관성, 접근성, 새로운 패턴, UX 요구 수준.
- 개발 보증인: 아키텍처, 보안, 성능, 유지보수성, 기술 품질, 운영 환경 조건.

원칙은 단순합니다.

누구나 여러 영역에 기여할 수 있지만, 각 영역에는 그 영역을 책임지는 누군가가 남아 있어야 합니다.

이 모델은 책임을 없애지 않고도 역할의 경계가 유연해지는 것을 허용합니다.

영업 담당자가 만들 수는 있어도 기술 보증인이 되는 것은 아니다

일부러 단순한 예시를 들어 보겠습니다.

영업 담당자가 필요를 발견하고 AI 지원 도구를 사용해 작은 애플리케이션이나 유용한 수정 사항을 만듭니다. 회사가 충분히 안전한 환경을 갖추었다면 직접 무언가를 만들 권리는 엔지니어링에만 있다고 볼 이유가 없습니다.

그러나 이러한 자유가 기술 품질까지 영업 담당자의 책임이 된다는 뜻은 아닙니다.

변경 사항이 보안과 권한, 민감한 데이터, 아키텍처에 영향을 미친다면 개발 보증인은 검토를 요구하고 수정을 지시하거나 변경을 막을 수 있어야 합니다.

영업 담당자는 자신의 필요를 계속 책임집니다.

엔지니어링은 회사가 기술적으로 유지하기로 받아들인 것을 계속 보증합니다.

디자인에도 같은 논리가 적용됩니다.

프로덕트 매니저는 알려진 컴포넌트를 조합해 화면을 만들 수 있습니다. 디자인 시스템과 익숙한 패턴 안에 머문다면 매번 디자인 검토를 요구하는 것은 자율성을 불필요하게 줄입니다.

반면 그 프로덕트 매니저가 디자인 시스템을 벗어나 새로운 패턴을 만들거나 구조적이고 복잡한 상호작용을 설계한다면 디자인 보증인이 개입할 수 있어야 합니다.

따라서 경계는 '누가 무엇을 건드릴 권리가 있는가?'가 아닙니다.

경계는 '어느 정도의 위험이나 새로움부터 해당 영역의 보증인이 다시 주도해야 하는가?'가 됩니다.

주제의 owner와 영역의 보증인은 같은 사람이 아니다

이 구분은 새로운 조직에서 가장 중요한 것 중 하나일 수 있습니다.

owner는 하나의 주제를 맡습니다.

문제를 이해하고, 작업을 진전시키고, 현장의 결정을 조정하고, 필요한 일을 조율하고, 결과를 확인하며, 주제가 결실을 맺도록 책임져야 합니다.

반면 보증인은 더 횡단적인 책임을 맡습니다.

보증인이 반드시 그 주제 자체를 맡는 것은 아닙니다. 한 영역의 일관성, 품질 또는 위험 수준을 맡습니다.

따라서 프로덕트 엔지니어는 한 기능의 owner가 될 수 있지만 회사 아키텍처 전체의 보증인이 되는 것은 아닙니다.

프로덕트 매니저는 출시의 owner가 될 수 있지만 모든 디자인의 보증인이 되는 것은 아닙니다.

영업 담당자는 업무 도구의 owner가 될 수 있지만 내부 애플리케이션 보안의 보증인이 되는 것은 아닙니다.

이 구분은 한 사람에게 모든 영역의 전문성을 요구하지 않으면서 결과에 대한 책임을 더 명확하게 만듭니다.

AMPECO는 이러한 움직임의 한 형태를 보여 줍니다. 프로덕트 오너와 엔지니어가 '무엇을'과 '왜'를 정렬한 뒤 에이전트가 분석과 개발, 테스트의 상당 부분을 맡습니다. 실행은 분산되지만 최종 검증의 책임은 인간에게 남습니다. [6]

따라서 AI는 하나의 주제를 실행하는 데 필요한 사람 수를 줄이는 동시에 누가 그것을 책임지는지 아주 명확하게 알아야 할 필요성을 키울 수 있습니다.

역할이 더 이상 적절한 분석 단위가 아닐 수도 있다

이 변화는 더 넓은 질문도 던집니다. 여전히 직책을 중심으로 회사를 조직해야 할까요?

프로덕트 오너를 둘러싼 논의는 이 문제를 잘 보여 줍니다. 저는 Malorean에서 PO란 백로그를 관리하기 때문에 존재하는 직업이라기보다, 의사결정과 책임의 권한을 부여받은 하나의 기능이라고 주장했습니다. [7]

이 구분은 AI와 함께 특히 유용해집니다.

수행하는 작업 목록으로 역할을 정의하면 AI로 인해 그 정의는 불안정해집니다. 많은 작업이 더 다양한 사람에게 열리기 때문입니다.

반면 한 사람이나 한 팀이 무엇을 책임져야 하는지로 책임을 정의하면 훨씬 더 잘 유지됩니다.

프로덕트 매니저가 명세를 덜 작성하더라도 프로덕트 선택의 품질을 책임지는 일은 멈추지 않습니다.

개발자가 코드를 직접 작성하는 양이 줄더라도 기술 품질을 책임지는 일은 멈추지 않습니다.

디자이너가 픽셀을 움직이는 유일한 사람이 아니게 되더라도 경험의 일관성을 보증하는 일은 멈추지 않습니다.

프로덕트, 디자인, 개발의 경계가 마찬가지로 덜 경직된 Cursor에서도 프로덕트 책임자는 본질이 그대로 남는다고 말합니다. 무엇을 만들 가치가 있는지 선택하고, 방향에서 벗어나지 않도록 충분한 지침을 제공하는 일입니다. [8]

따라서 직업의 변화는 모든 책임을 잃어서라기보다 특정한 실행 방식에 대한 독점권을 잃기 때문에 일어납니다.

보증인이 새로운 관료제가 되어서는 안 된다

하지만 명백한 함정이 하나 있습니다.

프로덕트 매니저가 코드를 건드릴 때마다 엔지니어링의 승인을 받아야 하고, 개발자의 모든 국지적 결정이 다시 프로덕트 매니저를 거쳐야 하며, 디자이너가 아닌 사람이 만든 모든 화면이 디자인 위원회를 기다려야 한다면 우리는 아무것도 해결하지 못한 것입니다.

과거의 필수 관문을 새로운 이름으로 되살렸을 뿐입니다.

보증인의 승인이 매번 필요한 절차가 되어서는 안 됩니다.

보증인이 존재하는 목적은 바로 더 많은 자율성을 가능하게 하는 것입니다. 조직은 책임이 어디에 있는지, 어떤 경우에 보증인을 다시 불러야 하는지를 알고 있으므로 각자는 끊임없이 허락을 구하지 않고도 알려진 틀 안에서 행동할 수 있습니다.

현실에서 이 모델이 완벽하게 작동하는 경우는 없습니다.

보증인이라면 정확히 그런 방식으로 승인하지 않았을 일이 운영 환경에 들어가기도 합니다. 국지적인 결정이 눈에 띄지 않고 지나가기도 합니다. 팀이 예상보다 더 많은 자유를 행사할 때도 있습니다. 반대로 조심하느라 너무 일찍 에스컬레이션하는 사람도 있습니다.

그렇다고 반드시 모델이 실패한 것은 아닙니다.

진정으로 자율적인 조직은 일부 결정이 사전 검사 없이 이루어진다는 사실을 받아들입니다. 그렇지 않다면 자율성이 아닙니다.

따라서 해결해야 할 문제는 어떤 어긋남도 일어나지 않게 보장하는 것이 아닙니다.

문제는 어떤 어긋남을 받아들일 수 있는지, 어떤 어긋남을 시스템이 감지해야 하는지, 어떤 어긋남이 큰 비용으로 이어지기 전에 보증인의 개입을 요구하는지 정의하는 것입니다.

바로 여기서 질문이 달라집니다.

우리는 두 가지 책임을 구분했습니다. 하나는 주제를 맡은 사람의 책임이고, 다른 하나는 한 영역의 보증인이 지는 책임입니다.

이제 또 다른 질문에 답해야 합니다.

언제 **owner**가 혼자 결정하도록 두고, 언제 보증인이 개입해야 할까요**?**

Malorean Corp.에서는 주도권과 책임 사이의 이 경계가 마찰을 일으키기 시작했습니다. 다음 이야기에서 그들을 다시 만난 뒤, 안전장치와 에스컬레이션 규칙을 살펴보겠습니다.

참고문헌

[1] Are Roles and Responsibilities a Thing of the Past? – Figma

- URL: https://www.figma.com/blog/2025-shifting-roles-report/
- 현대적인 제품 개발팀에서 책임의 범위가 넓어지고 직군 사이의 경계가 점차 유연해지는 현상을 연구합니다.

[2] PM, développeurs et IA : les rôles se brouillent, les responsabilités restent – Malorean

- URL: https://malorean.net/articles/2026-08-01-pm-developpeurs-et-ia-les-roles-se-brouillent-les-responsabilites-restent.html
- AI는 행동할 수 있는 사람의 범위를 넓히지만 프로덕트, 디자인, 엔지니어링의 책임까지 없애지는 않는다는 저자의 관찰을 발전시킨 글입니다.

[3] Designers' workflow for shipping code – Atlassian

- URL: https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code
- 디자이너가 제품 코드를 직접 수정하면서도 품질과 성능, 운영 배포를 위해 엔지니어링 검토를 유지하는 방식을 설명합니다.

[4] Product Engineer – Linear Careers

- URL: https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- 문제 정의부터 기능 출시에 이르기까지 Linear가 프로덕트 엔지니어에게 기대하는 바를 명시한 채용 공고입니다.

[5] La qualité appartient à ceux qui livrent – Malorean

- URL: https://malorean.net/articles/2026-06-03-la-qualite-appartient-a-ceux-qui-livrent.html
- 품질은 출시 이후 수리를 담당하는 팀에 넘길 수 없다는 명시적인 기술 책임 원칙을 주장합니다.

[6] How we built an AI-native engineering system – AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- 의도와 아키텍처, 검증은 인간의 책임 아래 두면서 실행의 상당 부분을 에이전트에게 맡기는 방식을 보여 줍니다.

[7] Le PO n'est pas un métier, c'est une fonction – et l'IA n'y change rien, elle accélère – Malorean

- URL: https://malorean.net/articles/2026-08-03-le-po-nest-pas-un-metier-cest-une-fonction.html
- 의사결정과 ownership의 권한을, 주로 산출물이나 백로그 관리로 정의되는 직업과 구분합니다.

[8] How Cursor Is Navigating the New Shape of Product Craft – Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- 역할의 경계가 더 유연해져도 프로덕트 판단과 목표, 안전장치가 핵심 책임으로 유지되는 모습을 보여 줍니다.

7 – Malorean Corp. – Alice가 글로 남기지 않고도 알고 있던 것

목요일 아침, Lucas는 그 주 들어 세 번째로 Alice에게 전화를 걸었습니다.

— 2분이면 돼요. 그가 약속했습니다.

Alice는 이 말을 경계하는 법을 배웠습니다. Lucas는 긴 회의를 요청하는 법이 없었습니다. 정확한 질문 하나를 들고 와서 함께 이야기한 뒤 다시 일하러 갔습니다. 바로 그 점 때문에 이런 방해를 거절하기 어려웠습니다.

— 말해 봐.

— 로트 추적 건인데요. 고객이 공급업체별로 검색하고 싶어 해요. 시연에는 추가할 수 있어요. 그런데 공급업체가 같은 내부 번호로 여러 로트를 납품했다면 결과를 한데 보여 줄까요, 아니면 하나를 선택하게 할까요?

Alice는 몇 초 동안 생각했습니다.

— 여러 개가 일치한다면 하나의 로트를 식별한 것처럼 보여서는 절대 안 돼. 결과를 분리해서 보여 줘. 고객이 어떻게 구분하는지는 함께 확인하면 되고.

— 저도 그렇게 생각했어요. 하지만 확신이 없었어요.

— 그럼 왜 내게 전화했어?

Lucas는 멋쩍게 웃었습니다.

— Julie의 목업 건 이후로는 프로덕트 규칙처럼 보이는 결정을 내리기 전에 확인하는 편이 낫겠다 싶어서요.

지난주 Alice는 잠재 고객 한 곳의 요청을 표준 기능으로 바꾸지 말라고 했습니다. 이제 Lucas가 그 대화에서 조금이라도 의심스러우면 자신의 의견부터 물어야 한다는 교훈을 얻었다는 사실이 아쉬웠습니다.

— 알겠어. 하지만 이 문제에는 내가 필요하지 않았어.

— 그렇게 말씀해 주시니 안심이 되네요.

두 사람은 통화를 마쳤습니다. 통화는 7분 동안 이어졌습니다. Alice는 작성하던 문서로 돌아가 마지막 문장을 다시 읽었지만 무엇을 더 쓰려 했는지 기억나지 않았습니다.

10분 뒤 Lucas가 메시지를 보냈습니다.

다른 질문이 있어요. 로트 정보가 없는 과거 제조 건에는 '알 수 없음'이라고 표시할까요, 아니면 열을 숨길까요?

Alice는 답을 쓰기 시작하다가 멈추었습니다.

질문은 달랐지만 판단 원리는 같았습니다. ERP에 없는 정보를 확실한 것처럼 보여 주어서는 안 된다는 것입니다. 두 사례를 따로 결정하는 대신 이 원칙을 알려 줄 수도 있었습니다.

그럼에도 Alice는 답했습니다.

정보가 없다는 사실을 명확히 표시해. 문제를 숨기지 말고.

Lucas는 엄지를 든 이모티콘으로 반응했습니다.

Alice는 일정을 살펴보았습니다. 지원팀과의 논의 하나가 미뤄져 있었고 오후에는 고객 미팅이 있었습니다. 그사이에 여러 사업장을 가진 잠재 고객을 위해 Julie가 만든 제안을 검토해야 했습니다.

Alice는 Lucas가 자신과 직접 이야기하는 것을 좋아했습니다. 그에게 주제를 맡기며 원했던 것이 바로 그것이었습니다. 하지만 Lucas가 너무 빨리 움직이다 보니 새로운 가능성이 생길 때마다 새로운 질문도 나타났습니다. 그가 자주 물을수록 Alice는 자신이 가장 잘할 수 있는 일, 즉 고객을 이해하고 제품의 방향을 고민하는 데 쓸 시간이 줄었습니다.

Lucas도 시간을 잃었습니다. 때로는 Alice의 답을 기다리느라 일을 재개하지 못했습니다.

불쾌한 생각 하나가 스쳤습니다. 이대로라면 결국 모든 결정을 다시 티켓 안에 넣게 될지도 몰랐습니다. 그러면 관리하기는 더 쉬울 것입니다. Alice

가 명세하고 Lucas가 실행하면, 누구도 자율성의 범위가 어디까지인지 고민할 필요가 없어집니다.

그들이 벗어나려 했던 바로 그 방식이었습니다.

정오에 Alice는 Bob과 화상 통화로 만났습니다. 두 사람은 의제 없이 대화하며 자신들에게 벌어지는 일을 비교하는 습관이 생겼습니다.

— Lucas와는 어떻게 되고 있어? Bob이 물었습니다.

— 어떤 의미에서는 너무 잘되고 있어.

— 칭찬처럼 들리지는 않는데.

— 아주 빠르게 움직이고, 많은 걸 알아보고, 고객과도 이야기해. 하지만 혼자 결정할 수 있는 일까지 계속 내게 전화해.

— 그래서 대답해 줘?

— 당연하지.

— 그럼 전화하는 게 맞네.

Alice는 눈을 굴렸습니다.

— 고마워, Bob.

— 반쯤은 진심이야. 틀릴 위험을 감수하는 대신 5분 만에 네 답을 들을 수 있다면 왜 마다하겠어?

— 이런 식으로는 절대 제대로 일할 수 없으니까.

Bob이 사례를 하나 말해 보라고 했습니다. Alice는 아침에 받은 두 질문과 Julie와 나눈 대화, 그리고 최근 Lucas가 망설였던 일을 들려주었습니다. 다른 고객에게는 거의 쓸모가 없을 설정을 고객 한 곳이 요청했다면 추가해야 하는가 하는 질문이었습니다.

— 알겠어? Alice가 말했습니다. 언제나 같은 질문의 변형이야.

— 그래서 네 답은 뭔데?

— 상황에 따라 달라.

Bob은 미소를 지었습니다.

— 바로 그거야. 그런 답을 하니까 사람들이 계속 전화할 수밖에 없지.

Alice는 반박하려 했지만 Bob의 말이 맞았습니다.

— 아무렇게나 '상황에 따라 다르다'는 뜻은 아니야. 반복해서 나타나는 문제인지, 우선순위 고객에게 도움이 되는지, 모두가 쓰는 제품을 무겁게 만들지는 않을지, 그 해결책을 유지할 수 있을지 살펴봐…

Alice는 말을 멈췄습니다.

— 봐. Bob이 말했습니다. 방금 규칙의 일부를 알려 줬잖아.

— 규칙이라기보다는 사고방식에 가까워.

— 오히려 더 좋네. 아마 Lucas에게 전달해야 할 게 바로 그거겠지.

Alice는 말이 없었습니다.

수년 동안 Alice는 티켓을 작성해 왔습니다. 기대하는 동작과 예외 사례, 승인 기준을 적었습니다. 하지만 가능한 여러 동작 사이에서 하나를 고르는 데 사용하는 판단 근거는 거의 적지 않았습니다. 그 추론은 머릿속에 남거나, 주제가 출시된 뒤 회의 속에서 사라졌습니다.

— 이런 게 아마 수백 가지는 있을 거야. Alice가 말했습니다.

— 그렇겠지. 나도 마찬가지고.

— 그런데 한 번도 적어 둘 이유가 없었어.

— 이유는 있었지. 시간이 없었을 뿐이야. 그리고 '좋은 아이디어를 때로 거절하는 이유 설명하기'라는 제목의 티켓을 여는 사람은 없었고.

두 사람은 웃었습니다.

통화를 마친 Alice는 새 문서를 열고 *프로덕트 의사결정 원칙*이라고 썼습니다.

첫 문장을 썼다가 지우고 다른 문장을 시도했습니다. 15분 뒤에는 사내 정책처럼 보이는 한 문단이 생겼습니다. 내용은 옳았지만 Bob에게 설명한 방식과는 전혀 달랐습니다.

Alice는 문서를 닫았습니다.

그리고 ChatGPT를 열었습니다.

— 우리 제품에 관해 내가 알고 있는 것을 머릿속에서 꺼낼 수 있게 도와줘. Alice가 받아쓰게 했습니다. 이론적인 절차를 원하지는 않아. 질문을 던지고, 사례를 떠올리게 하고, 진짜 규칙과 단순한 내 습관을 구분할 수 있게 도와줘.

첫 답변은 최근 내린 결정부터 시작하자고 제안했습니다.

Alice는 이야기하기 시작했습니다.

Julie의 잠재 고객과 매력적인 목업, 특별한 필요를 모든 고객을 위한 약속으로 바꿀 위험을 설명했습니다. 대기업 고객이 많은 돈을 가져올 수 있어도 자동으로 올바른 제품 방향이 되는 것은 아니라고 말했습니다.

— 잠깐, 그 이유부터 설명해야겠어. Alice는 소리 내어 덧붙였습니다. 우리 ERP는 서로 아주 다른 기업들이 사용해. 각자의 특별한 규칙을 소프트웨어 핵심에 모두 추가하면 결국 모두의 사용을 복잡하게 만들게 돼. 하지만 특별한 요청이 훨씬 일반적인 문제를 드러낼 때도 있어. 무조건 거절해서도 안 돼.

ChatGPT는 그 차이를 어떻게 구분하는지 물었습니다.

Alice는 지원팀과의 대화, 여러 고객에게서 관찰한 사례, 영업팀의 요청, 배포 제약을 설명했습니다. 기록을 전혀 남기지 않은 오래된 논의들도 떠올렸습니다. 왜 일부 선택지를 포기했는지, 왜 새로운 기능보다 설정을 택했는지, 왜 급여의 예외 규칙을 인터페이스 취향처럼 다룰 수 없는지도 설명했습니다.

때로는 명시적으로 물을 생각조차 해 본 적 없는 질문에 답하고 있는 자신을 발견했습니다.

AI는 내용을 다시 표현하고 묶어 하나의 원칙을 제안했습니다. Alice는 그것을 수정했습니다.

— 아니, '우리는 절대 고객 맞춤 개발을 하지 않는다'가 아니야. 특정 조건에서는 할 수 있지만, 그 비용이 무엇인지 의식적으로 결정해야 해. 무엇보다 영업 가설을 프로덕트 결정인 것처럼 넘겨서는 안 돼.

조금 뒤 ChatGPT는 추적 기능이 모든 과거 로트를 완전히 복원할 수 있다고 썼습니다.

— 절대 아니야. Alice는 화면을 보며 말했습니다. 데이터가 기록된 적이 없다면 마법처럼 복원할 수는 없어. 그건 약속할 역량이 아니라 명시해야 할 한계야.

Alice는 해당 내용을 수정하게 했습니다. 문서가 이미 결정된 것과 가능성이 높다고 생각하는 것, 그리고 지원팀이나 Lucas, 고객에게 확인해야 할 것을 구분하도록 요청했습니다. 실제 고객 계정의 기밀 정보는 넣지 않았고 사례는 다시 표현했습니다.

한 시간이 흐르고, 이어 두 시간이 흘렀습니다.

문서는 Alice가 오래전부터 알고 있던 내용으로 채워졌습니다. Malorean이 우선적으로 지원하려는 고객, 반복되는 문제, 받아들인 절충안, 하지 않기로 한 약속, 오래된 데이터의 한계, 프로토타입이 판매 가능한 기능과 같지 않은 이유가 담겼습니다.

이미 있는 정보로 가는 링크도 추가했습니다. 지원팀의 피드백, 아직 찾을 수 있는 결정, 미팅 메모, 제품 문서였습니다. 근거가 없는 곳에서는 그 사실을 밝혀야 했습니다.

세 시간이 거의 지났을 때 ChatGPT가 요약본을 제안했습니다.

Alice는 다시 읽었습니다. 어떤 페이지는 너무 길었고, 어떤 곳은 일반 원칙과 사례를 뒤섞었습니다. 지나치게 단정적인 표현을 마지막으로 수정하고 문서의 날짜와 범위를 추가했습니다.

그런 다음 전체를 PDF로 내보냈습니다.

Alice는 예상보다 많은 페이지를 넘겨 보았습니다.

세 시간 전만 해도 Lucas가 언제 자신 없이 결정할 수 있는지를 설명하려 했을 뿐입니다. 이제 손에는 팀이 제품에 관해 알고 있으면서도 한 번도 공유하지 않았던 것을 불완전하지만 놀랄 만큼 풍부하게 그려 낸 문서가 있었습니다.

최종적인 진실은 아니었습니다. 확인하거나 갱신해야 할 내용도 있었습니다. PDF가 고객과 데이터, 책임자의 판단을 대신하지도 않을 것입니다.

하지만 적어도 다음 사람은 처음부터 다시 시작하지 않아도 됩니다.

다음 날 아침 Lucas는 파일을 열었습니다.

— 이걸 전부 저를 위해 쓰신 거예요?

— 혼자 쓴 건 아니야. Alice가 대답했습니다. 세 시간 동안 ChatGPT와 이야기했어. 질문을 던지고, 내 답을 정리하고, 빠진 것을 찾는 데 도움을 받았지. 내가 다시 읽고 수정했고. 하지만 대부분은 이미 알고 있던 내용이야.

— 그럼 왜 저는 한 번도 본 적이 없죠?

Alice는 조금 멋쩍게 웃었습니다.

— 어떤 티켓에도 들어갈 내용이 아니었으니까.

Lucas는 목차를 살펴보았습니다. 원칙과 의사결정 사례, 업무 제약, 고객에 관한 정보, 출처 링크, 아직 답하지 못한 질문이 있었습니다.

— 추적 기능을 작업할 때 제 AI에 이걸 줘도 될까요?

— 승인된 환경 안에서는 괜찮아. 하지만 모든 것이 사실이라는 증거가 아니라 맥락으로 사용하라고 해. 중요한 규칙이라면 출처나 그 규칙을 책임지는 사람에게 다시 확인해야 하고.

— 엄청난 시간을 아낄 수 있겠어요. 가끔 AI에게 두 해결책 중 하나를 고르는 데 도움을 청하지만, AI는 제가 말해 준 것밖에 몰라요. 3년 전에 내린 결정을 전혀 고려하지 않은 채 아주 그럴듯한 답을 내놓을 수 있죠.

Alice도 그 느낌을 알았습니다. 따로 떼어 놓으면 합리적인 답이 그들의 ERP에는 터무니없을 수 있었습니다.

Lucas는 한 대목에서 멈췄습니다.

— 여기에는 누락된 업무 정보를 숨기면 결과가 완전해 보일 수 있으므로 절대 숨겨서는 안 된다고 쓰셨네요.

— 어제 네게 한 말이야.

— 네. 어제 이 문장이 있었다면 전화하지 않았을 거예요.

그는 계속 읽다가 고개를 들었습니다.

— 알겠어요. 하지만 아직 문제가 하나 있어요.

Alice의 얼굴에 낙담한 기색이 스쳤습니다.

— 뭔데?

— 이제 어떻게 판단하시는지는 더 잘 이해해요. 그래도 언제부터 혼자 결정해도 되는지는 모르겠어요.

Alice는 커피잔을 내려놓았습니다.

Lucas는 문서의 가장 큰 빈틈을 찾아냈습니다.

두 사람은 함께 문제를 다시 살펴보았습니다. Alice는 빈 페이지를 공유했습니다.

— 간단한 걸 시도해 보자. 내게 전화하기 전에 세 가지를 묻는 거야. 원하는 결과를 이해하고 있는가? 내 결정이 우리가 아는 틀 안에 머무는가? 틀렸을 때 무엇이 잘못될 수 있는가?

— '틀 안에 있다'는 말부터 정의해야겠네요.

— 바로 그거야.

두 사람은 세 단계를 적었습니다.

초록: 선택이 국지적인 범위에 머물고, 되돌릴 수 있으며, 알려진 규칙에 부합하고, 고객에게 새로운 약속을 하지 않는다면 Lucas가 혼자 결정합니다. 자신의 판단 근거를 설명하고 결과를 확인할 수 있어야 합니다.

주황: 새로운 상황, 목표 사이의 모순, 제품 동작에 관한 상당한 의문을 마주하면 Alice와 상의합니다. 기술적인 위험이라면 적절한 역량을 갖춘 사람에게도 도움을 요청합니다. ERP의 오래된 부분이 관련되었을 때 Thomas에게 묻는 경우가 그 예입니다.

빨강: 구조적인 규칙을 바꾸거나, Malorean을 대표해 상업적 약속을 하거나, 민감한 데이터를 다루거나, 되돌리기 어려운 중대한 위험을 감수하는 일은 누구도 혼자 결정하지 않습니다. 관련 책임자, 때로는 Hubert까지 참여해 명시적으로 판단해야 합니다.

Lucas는 기준표를 다시 읽었습니다.

— 그러면 혼동을 피하려고 시연에서 결과를 표시하는 방식을 바꾸는 건 초록이네요.

— 그래.

— 모든 고객에게 영향을 줄 수 있는 동작을 추가하고 싶다면 찾아오고요.

— 함께 논의해야지. 특히 필요를 제대로 이해하지 못했다면.

— 로트를 기록하려고 과거 데이터를 수정하고 싶다면요?

— 나 혼자 허락할 수 있는 일이 아니야. 기술 분석과 적절한 검사가 필요해. 네가 충분히 여러 번 상기시켜 줬잖아.

Lucas는 미소를 지었습니다.

— 기억하고 계신지 확인하고 싶었어요.

두 사람은 규칙을 하나 더 추가했습니다. Alice가 답하지 않았다고 해서 주황이나 빨강인 결정이 초록으로 바뀌지는 않습니다. 반대로 초록인 결정마다 Alice의 허락을 구해서도 안 됩니다.

— 색을 잘못 판단하면요? Lucas가 물었습니다.

— 내게 말해. 틀을 조정하면 돼. 내가 생각한 걸 네가 알아맞히지 못했다고 계속 탓하기보다 규칙을 개선하는 게 나아.

Lucas는 고개를 끄덕였습니다.

— 제 선택을 판단 근거와 빠진 정보까지 포함해서 이 기준표와 대조해 달라고 AI에 요청할 수도 있겠어요.

— 그래. 다만 어떤 위험을 받아들일 만한지는 AI가 우리 대신 결정하게 해서는 안 돼.

Alice는 기준표를 참고 문서에 추가했습니다. 정보를 전달하는 것과 실제로 결정할 수 있는 수단을 주는 것의 차이를 이해하기 시작했습니다.

여기서 멈출 수도 있었습니다. 하지만 Lucas는 추적 기능 시연으로 돌아갔습니다.

— 한 시간 동안 제 자율성에 관해 이야기했으니, 이제 주제 자체를 질문할게요. 무엇을 보면 이 일을 성공했다고 판단할 수 있을까요?

Alice는 거의 자동으로 대답했습니다.

— 고객이 특정 로트로 만든 제품을 찾아낼 수 있으면 되지.

— 맞아요. 하지만 어떻게 확인하죠? 어느 정도로 신뢰할 수 있어야 하고요?

이번에는 Alice가 곧바로 답을 찾으려 하지 않았습니다.

두 사람은 며칠 동안 화면과 규칙, 데이터 구조를 탐색했습니다. 가능성 있어 보이는 시연도 있었습니다. 그런데 결과를 어떻게 판단할지 정확히 적은 사람은 아무도 없었습니다.

Alice는 공급업체 경보를 받은 뒤 파일들을 대조하느라 하루 중 상당 시간을 보낸 품질 책임자를 떠올렸습니다.

— 거기서 다시 시작해야 해. Alice가 말했습니다. 경보가 발생한 로트 번호 하나로, 수작업 조사를 전부 반복하지 않고 관련 제조 건을 찾을 수 있는가? 얼마나 걸리는가? 결과는 완전한가? 정보가 없을 때 그 사실을 명확히 보여 주는가?

— 고객과 함께 시험할 수 있겠네요. 시연이 마음에 드는지만 물을 게 아니라요.

— 바로 그거야.

Alice는 새 페이지를 열었습니다.

— 사실은 시작하기 전에 이걸 먼저 했어야 했어.

Lucas는 웃음을 터뜨렸습니다.

— 2주 전에 그 말씀을 하셨다면 저를 다시 티켓 안에 넣으려 한다고 했을 거예요.

— 그랬을지도 모르지. 하지만 우리가 적고 있는 걸 봐.

두 사람은 관찰한 문제, 관련 사용자, 원하는 결과, 측정 방법, 데이터의 한계, 범위 밖에 둘 것, 검토가 필요한 결정을 적었습니다. 한 페이지를 넘기지 않았습니다.

8

Lucas는 이것을 *Mini PRD*라고 부르자고 제안했습니다.

— 어떤 화면을 만들라고 지시하는 게 아니네요. 그는 말했습니다. 무엇이 이 일을 쓸모 있게 만드는지 합의하고 있어요.

— 넘지 말아야 할 한계도 합의하고 있고.

Alice는 소리 내어 생각했습니다.

— 다음 주제부터는 이게 출발점이 되어야 해. 무엇을 개발할지 묻기 전에 어떤 결과를 얻고 싶은지, 어떻게 확인할지를 묻는 거야. 그런 다음에야 범위를 정하고 무엇을 만들 가치가 있는지 결정하는 거지.

— 제가 올바른 정보로 이 문서를 채울 수 있다면, 준비된 백로그를 기다리지 않고 시작할 수 있겠네요.

— 범위를 제안할 수 있다는 뜻이야. 아무 약속이나 검토 없이 승인한다는 뜻은 아니고. 하지만 그래, 그게 목적이야.

두 사람은 아직 결정할 수 없는 질문을 열린 상태로 두고 추적 기능의 Mini PRD를 완성했습니다. 일부 사례는 고객과 확인해야 했고 기술적인 제약도 확정해야 했습니다.

문서는 확실하지 않은 것을 확실한 것처럼 보이게 하지 않았습니다. 무엇을 배우려는지 밝혔습니다.

Alice는 확인란으로 이루어진 기능 목록보다 이 문서가 더 유용하다고 생각했습니다.

다음 월요일, Alice와 Hubert의 주간 미팅은 이전과 달랐습니다.

Alice는 평소의 진행 상황 표도, Lucas가 만든 기능 목록도 공유하지 않았습니다.

의사결정 기준표와 Mini PRD를 보여 주었습니다.

Hubert는 첫 부분을 읽었습니다.

— 이제 Lucas가 훨씬 빨라졌는데 문서를 만드는 데 이렇게 많은 시간을 쓴 건가요?

Alice는 이 반론을 알고 있었습니다. 며칠 전 자신도 똑같이 생각했습니다.

— 오히려 이런 문서가 부족했다고 생각해요. 무엇을 만들어야 하는지는 적었지만 선택에 필요한 정보는 거의 적지 않았어요. 그 결과 Lucas는 계

속 제게 결정을 요청했죠. 이게 있으면 일부 결정은 저 없이 내릴 수 있어요.

— 한계를 지키는지는 어떻게 알죠?

— 언제 혼자 결정할 수 있고 언제 상의해야 하는지 정했어요. 나머지는 전부 검사하지 않을 거예요. 결과를 확인하고 필요하면 틀을 고칠 겁니다.

Hubert는 오래된 데이터에 관한 문장에서 멈췄습니다.

— 기록된 적 없는 것까지 추적된다고 보장할 수는 없겠군요.

— 네. Alice가 대답했습니다. 문서에도 그렇게 적혀 있어요. 고객에게도 명확히 보여 줘야 하고요.

Hubert는 동의했습니다.

— 영업 담당자가 반대되는 약속을 하기 전에 알게 되어서 좋군요.

— 바로 그거예요. Alice가 말했습니다. 이 컨텍스트는 Lucas에게만 필요한 게 아니에요. Julie도 우리가 아는 것과 아직 결정하지 않은 것을 확인할 수 있어요. 다만 정보는 최신 상태로 유지해야 하고 어느 부분이 공식 기준인지도 모두 알아야 합니다.

두 사람은 수정 가능한 작업본을 유지하고 PDF는 날짜가 표시된 스냅샷으로 사용하기로 했습니다. 중요한 결정은 그 이유와 함께 기록합니다. 반복되는 원칙은 적절한 경우 에이전트 지침이나 도구의 검사 규칙으로 만들 수 있습니다. 그렇다고 운영 환경 배포 전에 필요한 기술 테스트나 검토가 사라지는 것은 아닙니다.

— 6개월 뒤 AI가 아무도 지지하지 않는 옛 규칙을 적용하고 있다는 걸 발견하고 싶지는 않아요. Hubert가 말했습니다.

— 저도요. Alice가 대답했습니다.

처음으로 Hubert는 Mini PRD에 가장 큰 관심을 보이는 듯했습니다.

— 고객이 실제로 로트를 찾아낼 수 있는지 확인할 건가요?

— 네. Alice가 대답했습니다. 함께 시험할 사례를 줄 거예요. 결과를 신뢰할 수 없다면 화면을 몇 개 완성했는지는 아무 의미가 없어요.

Hubert는 문서를 닫았습니다.

— 좋아요. 무엇을 출시했는지만이 아니라 무엇을 배웠는지 알고 싶습니다.

수요일, Alice는 메신저에서 이상한 점을 발견했습니다.

아침부터 Lucas가 보낸 메시지가 하나도 없었습니다.

프로젝트 채널을 확인해 보니 활동은 있었습니다. Lucas는 데모 인스턴스에서 결과를 표시하는 여러 방식을 시험했습니다. 그중 하나에서는 존재하지 않는 관계를 암시하지 않도록 같은 참조 번호를 가진 로트를 분리하기로 했다고 설명했습니다. 제품 맥락의 원칙을 인용하고, 사용한 데이터를 밝히고, 고객과 화면이 어떻게 이해되는지 확인하자고 제안했습니다.

그는 허락을 구하지 않았습니다.

Alice는 설명을 읽은 뒤 자기 일로 돌아갔습니다. 덧붙일 말이 없었습니다.

오후 늦게 Lucas가 전화를 걸었습니다.

— 주황인데 빨강으로 바뀔지도 모르는 게 있어요.

— 말해 봐.

— 고객이 과거의 제조 건에서도 검색하고 싶어 해요. 이력 중 일부에는 원자재와 완제품 사이에 신뢰할 만한 연결이 없어요. 우리가 아는 것은 보여 줄 수 있지만, 완전한 검색처럼 제시하면 검색 결과가 없다는 것이 관련 제품이 없다는 뜻이라고 생각할 수 있어요.

Alice는 자세를 바로잡았습니다.

— 맞아. 그런 모호함을 남겨서는 안 돼.

— 경고를 추가하고 데이터를 활용할 수 있는 기간을 구분할 수 있어요. 하지만 고객에게 무엇을 약속할지는 함께 결정하고 싶어요.

— 이건 그래야 해. 지원팀과도 이 한계를 어떻게 이해할지 확인해야 하고.

Lucas는 화면을 공유했습니다. 두 사람은 문제를 함께 다루기 시작했습니다. 버튼 하나를 승인하거나 Alice가 이미 아는 답을 전달하는 일이 아니었습니다. 소프트웨어에 대한 고객의 신뢰를 바꿀 수 있는 결정을 앞에 두고 있었습니다.

대화를 나누는 동안 Alice는 Lucas가 자신의 시간을 빼앗고 있다고 한 번도 생각하지 않았다는 사실을 깨달았습니다.

통화를 끝내기 전 Lucas가 망설였습니다.

— 있잖아요, 그 PDF가 제 모든 질문에 답해 주지는 않았어요.

— 다행이네. Alice가 말했습니다.

— 다행이라고요?

— 내가 모든 답을 담는 데 성공했다면 네가 더는 발견할 것이 없다는 뜻일 테니까.

Lucas는 미소를 지었습니다.

— 알겠어요. 그래도 언제 찾아와야 하는지는 더 잘 이해한 것 같아요.

통화가 끝난 뒤 Alice는 일정을 다시 열었습니다. 다음 고객 미팅까지 한 시간이 비어 있었습니다.

온전한 한 시간이었습니다.

이 실험이 더 큰 규모에서도 통할지, 진행 과정에서 얼마나 많은 실수를 할지는 아직 알 수 없었습니다. 다만 Lucas가 자신 없이 여러 결정을 내리며 앞으로 나아가다가, 자신의 판단이 정말 중요해지는 순간에 돌아왔다는 사실은 알았습니다.

Alice는 공유 문서에 과거 데이터를 표시하는 방법에 관해 명확히 해야 할 새 질문을 추가했습니다. Lucas에게 무엇을 코딩하라고 요구하는 티켓이 아니었습니다. 관련자들과 함께 살펴봐야 할 결정이었습니다.

그들의 업무 방식에서 요구 수준이 낮아진 것은 아닙니다.

요구 수준을 올바른 곳으로 옮기기 시작했을 뿐입니다.

8 – 더 많은 자율성에는 더 적은 틀이 아니라 더 많은 틀이 필요하다

조직은 더 많은 사람이 전문가를 기다리지 않고 행동하게 할 수 있습니다.

앞에서 살펴본 역할의 유연성과 책임은 새로운 조율 문제를 만듭니다. 이제 여러 사람이 같은 주제에 행동할 수 있지만, 반드시 같은 의사결정 기준을 가지고 있는 것은 아닙니다.

그렇다고 새롭게 생긴 행동 역량마다 인간의 체계적인 검토가 뒤따라야 한다면 자율성은 이론에 머뭅니다.

필수 관문을 한 직군에서 다른 직군으로 옮겼을 뿐입니다.

진짜 과제는 일관성과 보안, 품질, 프로덕트 방향을 잃지 않으면서 더 많은 결정을 현장에서 내릴 수 있는 시스템을 만드는 것입니다.

여기서 중요한 역설이 나타납니다.

자율성을 더 널리 나누고 싶을수록 틀은 더 명시적이어야 합니다.

자율성은 규칙이 없어서 생기는 것이 아닙니다.

목표와 한계, 위험, 책임이 충분히 명확하기 때문에 끊임없이 허락을 구하지 않고 행동할 수 있을 때 생깁니다.

모든 것을 검토받아야 한다면 자율성은 없다

하나의 주제를 책임진 프로덕트 엔지니어를 상상해 봅시다.

사용자와 이야기하고, 해결책을 탐색하고, 프로토타입을 만들고, 개발하고, 출시를 준비합니다.

문서상으로는 자율적입니다.

하지만 프로덕트의 사소한 판단마다 프로덕트 매니저에게, 인터페이스의 세부 사항마다 디자이너에게, 기술 결정마다 테크 리드에게 물어야 한다

면 조직은 책임을 실제로 옮긴 것이 아닙니다. 한 사람이 더 많은 승인 절차를 통과하게 했을 뿐입니다.

다른 직군도 같은 장애물을 만납니다. 누군가에게 만들 수 있는 수단을 주더라도 모든 행동에 과거의 프로세스와 승인이 필요하다면 얻는 것은 거의 없습니다.

따라서 질문은 누가 행동할 수 있는가에 그치지 않습니다.

어떤 틀 안에서 그 사람이 혼자 결정할 수 있는가를 알아야 합니다.

Cursor는 빌더가 비전이나 로드맵에서 벗어나지 않고 행동할 수 있을 만큼 명확한 목표와 안전장치를 중심으로 이 논리를 표현합니다. [1]

이때 조직은 상세한 지시의 일부를 의사결정 틀로 대체합니다.

티켓의 일부를 의사결정 틀로 대체하기

Linear의 사례를 통해 완전히 지시된 티켓이 왜 주도권을 제한할 수 있는지 살펴보았습니다. [2] 이제 질문은 실질적인 것이 됩니다. 새로운 명세를 기다리지 않고 팀이 스스로 판단하려면 어떤 정보를 가지고 있어야 할까요?

그렇다고 티켓을 없애야 하는 것은 아닙니다. 저는 Malorean에서 백로그를 프로덕트 전체의 기억이 아니라 행동과 조율을 위한 도구로 옹호했습니다. [3]

따라서 자율성의 틀은 모든 해결책을 미리 고정하지 않으면서 현장의 결정에 방향을 주는 정보를 명시할 수 있습니다.

이 틀에는 다음이 포함될 수 있습니다.

- 우선순위 고객 또는 ICP
- 해결할 문제
- 기대하는 결과
- 범위의 한계
- 중요한 제약

- 기대하는 품질 수준
- 조직이 하지 않기로 명시적으로 받아들인 것
- 넘어서면 결정을 에스컬레이션해야 하는 기준선

Ramp는 이 논리를 구체적으로 보여 줍니다. 핵심 ICP와 핵심 문제, 회사가 특히 잘 해결하려는 것을 분명히 하면 수익성 높은 기회라도 제품을 다른 방향으로 끌고 갈 때 거절할 수 있습니다. [4]

Figma는 0에서 1로 빠르게 가는 것만으로는 충분하지 않다고 상기시킵니다. 빠른 프로토타입을 좋은 제품으로 만들려면 실제 사례와 예외, 품질 수준이 여전히 결정적입니다. [5]

이 기준표는 조직의 한 가지 선택을 보여 줍니다. 무엇을 해야 하는지만 밝히는 것이 아니라 지켜야 할 한계도 보이게 하는 것입니다. 여기서 이를 보편적인 방법론으로 제시하려는 것은 아닙니다.

컨텍스트가 인프라가 된다

현장에서 좋은 결정을 내리려면 문제를 이해하게 해 주는 정보에 접근할 수 있어야 합니다.

이것은 아마 에이전트 시대의 가장 중요한 조직 변화 중 하나일 것입니다.

더 많은 사람에게 결정을 맡기면서 핵심 맥락을 소수 전문가의 머릿속에만 둘 수는 없습니다. Linear는 고객 피드백과 전략 방향, 결정, 코드에 인간과 에이전트 모두가 접근할 수 있어야 한다고 강조합니다. [2]

저는 Malorean에서 프로덕트 매니저를 컨텍스트 아키텍트로 설명했습니다. 인터뷰, 영업 과정의 반론, 고객 지원, 결정, 데이터, 출처, 약한 신호처럼 흩어진 지식을 재사용할 수 있는 조직 자산으로 바꾸는 역할입니다. [6]

이 아이디어는 문서의 기능을 근본적으로 바꿉니다.

목표는 더 많은 페이지를 만드는 것이 아닙니다.

누군가 모든 추론을 처음부터 다시 세우지 않고도 결정에 필요한 요소에 접근할 수 있게 하는 것입니다.

좋은 컨텍스트가 있으면 다음을 이해할 수 있습니다.

- 고객이 실제로 한 말
- 이미 시험한 가설
- 중요한 제약
- 이미 내린 구조적인 결정
- 이미 받아들인 절충안
- 결과를 판단할 수 있는 지표

자율성이 커질수록 이 컨텍스트는 더 신뢰할 수 있고 널리 공유되어야 합니다.

그렇지 않으면 회사는 행동할 역량만 나누고 이해할 역량은 나누지 않는 셈입니다.

'왜'를 기억하기

이러한 재분배는 또 다른 문제를 만듭니다.

더 많은 결정이 현장에서 내려집니다. 때로는 서로 다른 사람들이, 때로는 이전 판단에 참여하지 않았던 에이전트와 함께 결정합니다.

그런데 기억되지 않은 결정은 금세 다시 협상 대상이 됩니다.

결론은 기억하지만 맥락은 잊습니다.

규칙이 있다는 사실은 알지만 왜 선택했는지는 모릅니다.

새로운 고객이 오고, 새로운 관리자가 부임하고, 팀이 바뀌면 이미 결론 낸 논의를 처음보다 더 적은 정보로 다시 시작합니다.

제가 **Product Decision Record**에 부여하는 역할이 바로 이것입니다. [7]

PDR은 사소한 판단 하나하나를 기록하지 않습니다.

여러 팀과 제품, 미래의 여러 상황을 이끌 만큼 구조적인 프로덕트 결정을 기록합니다.

특히 다음을 보존합니다.

- 맥락
- 검토한 대안
- 채택한 결정
- 선택한 이유
- 긍정적·부정적 결과
- 받아들인 위험
- 결정을 다시 논의할 만한 조건
- 판단하거나 자문한 사람들

행정 업무를 늘리는 것이 목적은 아닙니다.

오히려 좋은 PDR은 같은 논의를 쉰 번 다시 여는 일을 막아야 합니다.

그러면 유용한 구분 하나가 생깁니다.

코드는 무엇을 만들었는지 기록하고, 의사결정의 기억은 왜 특정 구조적 규칙을 선택했는지 기록해야 합니다.

지속 가능한 자율성에는 둘 다 필요합니다.

안전장치를 시스템으로 옮기기

좋은 컨텍스트가 있어도 모든 직원에게 조직의 모든 규칙을 기억하라고 하는 것은 비효율적입니다.

그래서 일부 기업은 반복되는 규칙을 인간의 검토에서 도구로 옮깁니다. 이는 관찰할 수 있는 메커니즘일 뿐, 자동화만으로 품질을 보장할 수 있다는 뜻은 아닙니다.

Meta는 변경 사항이 장애를 일으킬 확률을 추정하고 위험에 민감한 워크플로를 운영하기 위해 Diff Risk Score를 사용합니다. [8]

Cloudflare는 명시적인 엔지니어링 표준을 자동화된 검사로 바꿉니다. 일부 필수 규칙은 변경이 수정되거나 명시적인 예외가 승인될 때까지 승인을 막을 수 있습니다. [9]

이 움직임은 전문가의 역할을 바꾸므로 중요합니다.

한 규칙을 자동으로 확인할 수 있다면 기술 보증인이 풀 리퀘스트 쉰 개에서 같은 규칙을 일일이 상기시킬 필요가 없습니다.

디자인 시스템이 올바른 선택을 쉽게 하고 잘못된 선택을 눈에 띄게 만든다면 디자인 보증인이 버튼 하나하나를 승인할 필요가 없습니다.

구조적인 결정과 타깃, 목표에 접근할 수 있다면 프로덕트 보증인이 같은 제약을 매주 다시 설명할 필요가 없습니다.

틀이 인프라가 됩니다.

테스트, CI, 디자인 시스템, 표준, 보안 규칙, 템플릿, 에이전트 지침, 지표, 공유 컨텍스트가 과거에 사람이 맡던 통제의 일부를 담당합니다.

그러면 인간의 전문성은 예외에 집중할 수 있습니다.

모든 결정에 같은 수준의 통제가 필요한 것은 아니다

바로 여기서 위험이라는 개념이 핵심이 됩니다.

텍스트 번역과 권한 시스템 수정이 같은 검토 절차를 따라서는 안 됩니다.

국지적이고 되돌릴 수 있는 변경에는 데이터 마이그레이션이나 회사의 포지셔닝을 바꾸는 결정과 같은 수준의 검토가 필요하지 않습니다.

이런 조직을 이해하는 한 가지 방법은 '검토가 필요한가?'라는 이분법적 질문을 단계적인 질문으로 바꾸는 것입니다. 이 위험에는 어느 정도의 통제가 필요한가**?**

다음 세 단계는 이 책에서 제안하는 분석 틀이며, 인용한 모든 기업에서 확인된 관행은 아닙니다.

현장 자율

결정이 알려진 목표와 표준 안에 머뭅니다.

되돌릴 수 있고, 가능한 결과가 제한적이며, 구조적인 규칙을 바꾸지 않습니다.

owner가 결정합니다.

표적 검증

결정이 한 영역에 새로운 요소나 예외, 상당한 위험을 들여옵니다.

개발, 디자인, 프로덕트 등 해당 영역의 보증인이 개입합니다.

보증인이 반드시 주제를 넘겨받는 것은 아닙니다. 부족한 특정 판단을 보탭니다.

에스컬레이션

결정이 시스템 자체의 경계를 바꿉니다.

기술 측면에서는 구조적 아키텍처, 보안, 민감한 데이터, 중대한 마이그레이션, 규제 위험이 해당합니다.

디자인 측면에서는 중요한 새 상호작용, 디자인 시스템과의 단절, 다른 사용자 흐름에 대한 중대한 영향이 해당합니다.

프로덕트 측면에서는 타깃, 가치 제안, 가격 정책, 대규모 투자, 로드맵, 전략적 판단의 변경이 해당합니다.

이 경우 owner는 더 이상 틀 안에서 결정하는 데 그치지 않습니다.

틀 자체를 바꾸자고 제안합니다.

여기서 제안하는 모델에서는 이런 상황에 보증인이나 더 높은 의사결정 단계가 필요합니다.

명시적인 한계가 없을 때 생기는 상반된 두 위험

의사결정의 한계가 명확하지 않으면 조직은 서로 반대되는 두 가지 위험에 노출됩니다.

첫 번째는 관료주의입니다.

모든 것이 위로 올라갑니다. 전문가는 대기열이 됩니다. 책임자는 사소한 결정을 승인해야 하고 자율성은 사라집니다.

두 번째는 책임의 희석입니다.

각자는 언제부터 결정이 국지적인 범위를 벗어나는지 모른 채 현장에서 결정합니다. 서로 모순되는 선택이 쌓이다가 고객이나 회사가 볼 수 있는 문제로 나타납니다.

저는 Malorean에서 이런 경계를 더 명확하게 만드는 여러 조직 일관성 도구를 설명했습니다. 공유 언어, 원페이지 문서, 지표의 위계, 의사결정 기록, 에스컬레이션 규칙입니다. [10]

그 가치는 더 많은 문서를 만드는 데 있지 않습니다.

다음 질문을 명시하게 만드는 데 있습니다.

- 누가 결정하는가?
- 어떤 주제에 관해 결정하는가?
- 언제 결정하는가?
- 무엇이 현장에 남을 수 있는가?
- 무엇이 상위 단계로 올라가야 하는가?
- 어떤 조건에서 결정을 다시 논의할 수 있는가?

이 두 위험은 하나의 조직 가설로 이어집니다. 검토를 늘리거나 없애는 것보다 결정을 더 올바른 위치에 두는 것이 중요할 수 있습니다.

결정은 손실을 받아들이는 일이기도 하다

안전장치가 기술적인 것에만 머물러서는 안 됩니다.

진짜 선택을 내릴 수 있는 역량은 틀의 핵심입니다.

조직은 종종 합의가 이루어진 듯 보일 만큼 모호한 결정을 만듭니다. 각자가 거기에 자신의 우선순위를 투영하고, 실행 단계에 가서야 제약들이 서로 양립할 수 없었다는 사실을 발견합니다.

저는 이를 어정쩡한 결정이라고 부릅니다. [11]

자율성이 매우 높은 시스템에서 특히 위험합니다.

목표가 모호하면 각 팀이 현장에서는 합리적이지만 전체적으로는 일관되지 않은 결정을 내릴 수 있습니다.

따라서 더 많은 자율성을 주려면 받아들인 손실도 보이게 해야 합니다.

어떤 고객이 우선인가?

지금은 어떤 세그먼트를 지원하지 않을 것인가?

어느 품질 수준만큼은 양보할 수 없는가?

어떤 제약은 일시적으로 희생할 수 있는가?

두 목표가 충돌할 때 어느 목표가 우선하는가?

자율성에 모든 것을 말하는 틀이 필요한 것은 아닙니다.

현장의 결정이 같은 방향으로 모일 만큼 진짜 선택을 분명하게 만드는 틀이 필요합니다.

결과 측정이 마지막 안전장치가 된다

자율성은 무언가를 출시하는 순간 끝나지 않습니다.

한 사람이 실제로 주제를 맡았다면 그 뒤에 일어나는 일도 살펴봐야 합니다.

첫 장에서는 이미 생산량과 창출한 가치를 구분했습니다. 여기서 주제는 다릅니다. 결정을 내린 사람은 그 결정이 적절했는지 나중에 어떻게 알 수 있을까요?

Canva는 구체적인 메커니즘을 제공합니다. 팀이 실험을 예상 사업 효과와 연결할 수 있게 합니다. [12]

각 주제에서 다음을 명시하게 됩니다.

- 기대한 변화
- 그 변화를 관찰할 수 있는 측정값
- 지켜야 했던 품질
- 사용자가 실제로 한 행동
- 계속할지, 수정할지, 포기할지, 더 깊이 살펴볼지

측정은 책임의 순환을 닫습니다.

측정이 없으면 자율성은 단순히 생산해도 된다는 허가가 될 수 있습니다.

측정이 있으면 결과에 대한 책임이 됩니다.

자율성은 경영진의 후퇴가 아니다

1.4장에서 소개한 Marty Cagan의 자율적인 팀 모델은 경영진의 역할을 더 분명히 하는 데 도움이 됩니다. 미리 정한 해결책 없이 문제를 팀에 맡기려면 목표와 전략적 맥락을 명확히 해야 합니다. [13] Cagan은 역량 개발과 지원, 리더십도 강조합니다. 팀이 결정하게 두는 것은 팀을 방치하는 일이 아닙니다. [14]

다만 이 비교에는 중요한 한계가 있습니다. Cagan은 서로 보완하는 역량을 가진 프로덕트 팀의 책임을 설명합니다. 이 책이 제안하는 보증인 개념은 다른 문제에 답합니다. 실행이 한 직군에서 다른 직군으로 넘어갈 때 한 영역의 책임을 어떻게 보존할 것인가 하는 문제입니다.

보증인은 모든 행동의 승인자가 아니라 틀을 지키는 사람이 된다

1.6장은 모든 작업을 직접 실행하지 않더라도 한 영역의 일관성을 책임지는 사람으로 보증인을 소개했습니다. 여기서 그 역할은 구체적이 됩니다. 이해할 수 있는 규칙을 정하고, 반복되는 검사를 자동화하며, 위험이나 새로움이 팀이 혼자 결정할 수 있는 범위를 넘어설 때 개입하는 것입니다.

모든 것을 검사해야 한다면 보증인은 다시 병목이 됩니다. 아무도 아무것도 책임지지 않는다면 책임은 희석됩니다.

따라서 자율성은 모든 행동을 검토하는 것도, 모든 통제를 없애는 것도 아닙니다. 명확한 책임을 유지하면서 더 많은 현장 결정을 가능하게 하는 것입니다.

그러나 이러한 변화는 프로덕트와 엔지니어링 직군만을 넘어서는 질문을 낳습니다.

컨텍스트가 직접 전달되고, 결정을 더 현장에서 내릴 수 있으며, 에이전트가 조율의 일부를 흡수한다면 과거에 이런 기능을 맡았던 관리 계층은 어떻게 될까요?

먼저 Malorean Corp.에서 이 질문이 구체적인 사람들에게 어떻게 나타나는지 보겠습니다. 그런 다음 회사 전체에 어떤 의미가 있는지 살펴보겠습니다.

참고문헌

[1] How Cursor Is Navigating the New Shape of Product Craft – Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- 명확한 목표와 안전장치, 프로덕트 판단이 역할의 경계가 유연한 사람들도 선택한 방향에서 벗어나지 않고 행동하게 하는 방식을 보여 줍니다.

[2] Issue tracking is dead – Linear

- URL: https://linear.app/next
- 고객 피드백과 전략, 결정, 코드가 티켓 중심의 연쇄 구조보다 더 많은 컨텍스트를 공유하는 시스템을 옹호합니다.

[3] Le backlog n'est pas un dépotoir : c'est un outil d'action – Malorean

- URL: https://malorean.net/articles/2026-06-03-le-backlog-nest-pas-un-depotoir.html
- 일시적인 행동·조율 도구인 백로그를 장기적인 프로덕트의 기억, 컨텍스트, 추론과 구분합니다.

[4] 5 insights from Ramp's CPO on building for hypergrowth – Ramp

- URL: https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- 핵심 ICP와 핵심 문제를 강조하고, 제품을 분산시키는 기회를 거절해야 할 필요성을 설명합니다.

[5] What It Takes to Build Great Products Now – Figma

- URL: https://www.figma.com/blog/what-it-takes-to-build-great-products-now/
- 빠른 프로토타입만으로는 충분하지 않으며 지속 가능한 제품에는 실제 사례와 예외, 품질이 여전히 결정적이라고 상기시킵니다.

[6] Le PM comme architecte du Contexte – Malorean

- URL: https://malorean.net/articles/2026-06-08-le-pm-comme-architecte-du-contexte.html
- 흩어진 신호와 출처, 결정, 판단을 재사용할 수 있는 컨텍스트로 바꾸어 미래의 의사결정을 개선하자고 제안합니다.

[7] Product Decision Record : tracer les choix produit qui structurent l'entreprise – Malorean

- URL: https://malorean.net/articles/2026-06-03-product-decision-record-tracer-les-choix-produit.html
- 구조적인 결정의 맥락과 대안, 선택, 결과, 명시적인 재검토 조건을 담는 짧은 기억을 형식화합니다.

[8] Diff Risk Score: AI-driven risk-aware software development – Meta

- URL: https://engineering.fb.com/2025/08/06/developer-tools/diff-risk-score-drs-ai-risk-aware-software-development-meta/
- 소프트웨어 변경이 장애를 일으킬 확률에 따라 통제 워크플로를 조정하는 위험 점수를 소개합니다.

[9] How Cloudflare enforces engineering standards using AI – Cloudflare

- URL: https://blog.cloudflare.com/engineering-standards-enforcement/
- 명시적인 표준이 운영 환경 배포 전에 특정 위반을 막을 수 있는 자동 검사로 바뀌는 방식을 보여 줍니다.

[10] Les outils de cohérence organisationnelle – Malorean

- URL: https://malorean.net/articles/2026-06-02-les-outils-de-coherence-organisationnelle.html
- 공유 언어와 목표, 지표, 의사결정의 기억, 에스컬레이션 규칙을 연결해 조직의 판단을 명시적으로 만듭니다.

[11] Pourquoi les organisations préfèrent les décisions molles – Malorean

- URL: https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.html
- 손실을 가리고 책임을 희석하며 모순을 실행 단계로 미루는 합의형 비결정을 분석합니다.

[12] Measuring Commercial Impact at Scale – Canva

- URL: https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- 현장의 실험과 프로덕트 결정, 사업 가치 창출을 연결하는 셀프서비스 사업 효과 측정을 설명합니다.

[13] Team Objectives - Empowerment – Marty Cagan, Silicon Valley Product Group (2020)

- URL: https://www.svpg.com/team-objectives-empowerment/
- 현장의 결정에 필요한 전략적 맥락과 함께 해결할 문제를 팀에 맡겨야 한다고 주장합니다.

[14] Product Leadership Is Hard – Marty Cagan, Silicon Valley Product Group (2020)

- URL: https://www.svpg.com/product-leadership-is-hard/
- 자율적인 프로덕트 팀에는 경영진의 후퇴가 아니라 더 높은 수준의 리더십과 적극적인 지원이 필요한 이유를 설명합니다.

9 – Malorean Corp. – 합리적인 두 결정

월요일 아침, 두 개발팀의 조율을 책임지는 Vivien은 수년 동안 해 온 것과 같은 질문으로 회의를 시작했습니다.

— 이번 주 주제들이 어디까지 진행됐는지 말해 줄 사람 있나요?

그의 표는 이미 열려 있었습니다. 티켓과 담당자, 마감일, 장애물이 오랫동안 공들여 다듬은 열에 정리되어 있었습니다. 그는 각 다섯 명으로 이루어진 두 팀과 그들의 강점, 습관, 더 재촉해야 할 주제, 익을 때까지 두는 편이 나은 주제를 잘 알았습니다.

아무도 바로 답하지 않았습니다.

잠시 뒤 Lucas가 말했습니다.

— 추적 기능은 금요일에 Mini PRD를 갱신했어요. 고객과 로트 검색을 시험했고, 다음 단계는 데이터가 불완전한 사례를 검증하는 거예요. 열린 질문은 컨텍스트에 적어 뒀습니다.

— 출시일은요? Vivien이 물었습니다.

— 과거 재고 검사가 끝나야 알 수 있어요. Thomas가 저와 함께 한 가지를 살펴봐야 해요.

Thomas는 고개를 끄덕였습니다. 그는 출하 준비 흐름을 개선하고 있었고 도구에 진행 상황도 이미 갱신했습니다.

Vivien은 다른 개발자들에게 넘어갔습니다. 대답은 비슷했습니다. 모두가 자신의 문제를 더 잘 이해하고 있었고, 몇 가지 결정을 내렸으며, 다음 단계를 설명할 수 있었습니다. 여전히 일을 나누는 데 도움을 요청하는 사람도 있었지만, 나머지는 회의를 기다리지 않고 진전했습니다.

마지막에 Vivien은 자신의 표를 바라보았습니다.

방금 들은 내용이 이미 모두 적혀 있었고, 때로는 더 자세했습니다.

— 좋아요. 다 살펴봤네요.

회의는 12분 만에 끝났습니다.

전에는 45분이 걸렸을 것입니다.

Vivien은 기뻐해야 할지 확신하지 못했습니다.

그는 개발자들이 모두 르아브르 사무실에 모여 일하던 시절부터 Malorean에서 근무했습니다. 출시를 조직하고, 지연을 관리하고, 프로덕트 팀과 우선순위를 협상하고, 금요일까지 된다고 했던 요청이 왜 다음 화요일에나 나올 수 있는지 대표에게 설명했습니다.

코로나19 이후 회사가 원격 근무로 전환하자 방법을 바꾸었습니다. 복도에서 나누는 대화만으로는 부족했습니다. 결정을 기록하고, 장애물을 보이게 하고, 다른 부서와 합의한 내용을 모두가 적도록 요구했습니다.

그것이 일상이 되었습니다. 누구도 한눈에 보지 못하는 정보를 흐르게 하는 일이었습니다.

Vivien은 그 일을 좋아했습니다. 특히 잘못된 길로 가던 프로젝트가 누군가 적절한 사람들을 연결한 덕분에 다시 길을 찾는 순간을 좋아했습니다.

하지만 팀 업무에 AI 도구가 들어온 뒤 무언가 달라졌습니다. 개발자들이 더 빨리 생산하게 된 것보다 더 중요한 것은 일부가 주제를 다르게 다루기 시작했다는 점이었습니다. Lucas는 고객과 직접 이야기했습니다. Alice는 개발할 기능 목록뿐 아니라 결정의 이유를 설명하는 프로덕트 컨텍스트를 공유했습니다. Bob도 자기 주제에서 같은 방식을 도입하기 시작했습니다.

이제 에이전트는 Vivien이 전에는 직접 만들던 요약의 일부를 준비했습니다.

일이 사라진 것은 아니었습니다. 조각조각 그의 손을 떠나거나 다른 곳에서 이루어졌습니다.

그는 아직 아무에게도 이 이야기를 하지 않았습니다.

다음 수요일, Lucas는 Alice에게 추적 기능 시연의 새 버전을 보여 주었습니다.

— 공급업체가 결함 가능성이 있는 로트를 신고했을 때를 시험했어요. 우리가 알고 있는 제조 건은 찾을 수 있어요. 그리고 의심이 전혀 없는 로트처럼 보이지 않도록 '검토 중' 상태를 추가했어요.

Alice는 화면을 살펴보았습니다.

상황을 검토해야 한다는 배너가 표시되었습니다. 오래된 제조 건에는 일부 정보를 사용할 수 없다는 두 번째 메시지가 나타났습니다.

— 첫 번째 버전보다 솔직하네. Alice가 말했습니다.

— 네. 고객사의 품질 책임자가 '확인된 문제가 없음'과 '필요한 정보가 없음'을 섞지 말아 달라고 했어요.

Alice는 동의했습니다. 컨텍스트 문서에서 이미 논의한 구분이었습니다.

— Mini PRD에 적었어?

— 네. 테스트 사례도 추가했어요.

— 그럼 계속해.

Alice는 배너의 위치나 메시지의 정확한 문구를 결정할 필요가 없었습니다. Lucas는 어떤 결과를 얻으려는지 알고 있었습니다. 표시 방식을 하나 선택했고 고객과 확인할 계획도 세웠습니다.

조금 뒤 Thomas는 Vivien에게 출하 모듈 개선안을 보여 주었습니다.

문제는 달랐습니다. 일부 고객사에서는 명시적으로 금지된 사항이 없어도 주문이 중간 상태나 잘못 입력된 상태 때문에 멈추곤 했습니다. 물류팀은 확인된 위험이 없는 서류를 수정하게 하느라 시간을 잃었습니다.

Thomas는 준비 규칙을 명확히 했습니다. '차단'으로 표시된 로트는 출하할 수 없고, 다른 상태는 계속 표시하되 기존 검사를 거쳐 평소 흐름을 따릅니다.

— 다른 모듈의 동작은 바꾸지 않는 게 확실해요? Vivien이 물었습니다.

— 기존 사례에서는요. 회귀 테스트도 추가했습니다. 차단 규칙은 건드리지 않았어요.

Vivien에게 Thomas의 신중함을 의심할 이유는 없었습니다. Thomas는 오래된 문서에도 더는 나오지 않는 예외를 알고 있을 때가 있었습니다.

— 좋아요. 더 진행하기 전에 변경 사항을 검토받으세요.

Thomas는 고개를 끄덕였습니다.

두 변경 사항은 아직 하나의 시나리오에서 함께 시험되지 않았습니다.

문제는 금요일 통합 테스트 인스턴스에서 나타났습니다.

지원팀 직원이 원자재 입고, 제조, 공급업체 경보, 관련 제품 식별, 출하 준비까지 이어지는 전체 흐름을 준비했습니다.

Lucas는 화상 회의로 시연을 지켜보았습니다. Thomas도 참석했습니다. Alice는 미래의 동작을 설계에 참여하지 않은 사람이 어떻게 이해하는지 보고 싶어 초대를 요청했습니다.

처음 몇 분은 모두 순조로웠습니다.

로트 번호를 찾고 제조 건이 나타났으며 '검토 중' 배너도 표시되었습니다.

그런 다음 지원팀 직원이 출하 모듈로 이동했습니다.

출하 전표는 준비 상태에 머물렀습니다. 진행을 막는 메시지는 없었습니다.

— 잠깐만요. Alice가 말했습니다. 같은 로트 이야기 맞죠?

Lucas는 화면 쪽으로 몸을 기울였습니다.

— 네. 왜요?

— 위에서는 검토 중인데, 여기서는 아무 일도 없는 것처럼 계속할 수 있잖아.

Thomas는 주문 상세를 열었습니다.

— 차단 상태가 아니라 검토 중이에요. 같은 상태가 아닙니다.

— 제게 '검토 중'은 누군가 판단하기 전까지 승인된 것으로 보면 안 된다는 뜻이었어요. Lucas가 대답했습니다.

Thomas는 카메라를 바라보았습니다.

— 그럼 왜 '차단'이라고 표시하지 않았죠?

— 일반적인 금지 규칙을 만들어 내고 싶지 않았으니까요. 모든 활동을 멈추지 않고 조사해야 하는 고객도 있어요. 실제로 무엇이 관련되었는지는 그 회사의 품질 책임자가 결정해야 해요.

— 제 모듈에는 정보가 불완전한 오래된 로트가 있어요. Thomas가 반박했습니다. 명시적으로 승인되지 않은 것을 전부 막기 시작하면 어떤 주문은 영원히 진행되지 않아요.

Alice는 두 화면을 바라보았습니다.

Lucas의 이유도 이해했고 Thomas의 이유도 이해했습니다.

문제는 사용자가 ERP 한쪽에서는 '검토 중'이라는 문구를 보면서 다른 쪽에서는 출하를 막는 경고를 전혀 만나지 않을 수 있다는 것이었습니다.

— 시연은 여기서 멈춰요. Alice가 말했습니다.

아무도 이의를 제기하지 않았습니다. 실제 고객 데이터에는 손대지 않았고 변경 사항도 운영 환경에 배포되지 않았습니다. 테스트가 바로 이런 일을 위해 존재했습니다.

하지만 뒤이은 침묵은 성공적인 시연 뒤의 침묵보다 불편했습니다.

Vivien은 방금 벌어진 일을 전해 듣고 통화에 들어왔습니다.

Lucas가 자신의 결정을 설명하고 Thomas가 자기 결정을 이야기할 때까지 기다렸습니다. 테스트와 각 변경의 범위에 관해 몇 가지 질문을 던졌습니다.

— 각자 Mini PRD가 있나요? 그가 물었습니다.

— 네. Lucas가 대답했습니다.

— 의사결정 규칙도요?

— 네. Thomas가 대답했습니다. 기존 차단 동작은 바꾸지 않았어요. 준비 흐름의 국지적인 개선이었습니다.

— 저도요. Lucas가 말했습니다. 의문을 숨기지 않도록 정보를 추가했어요. 바로 그게 중요하다고 적어 두기도 했고요.

Vivien은 두 문서를 나란히 열었습니다.

어느 쪽도 틀리지 않았습니다. 단지 같은 문제를 말하고 있지 않았습니다.

첫 문서는 추적 과정의 불확실성을 어떻게 알릴지 설명했습니다. 두 번째 문서는 언제 출하를 멈출지 설명했습니다. 의심스럽지만 명시적으로 차단되지는 않은 로트가 있을 때 소프트웨어가 무엇을 해야 하는지, 두 문서를 연결하는 질문에는 어느 쪽도 답하지 않았습니다.

— 두 주제 사이에 구멍이 있군요. Vivien이 말했습니다.

Thomas는 동의했습니다.

— 기술적으로는 여러 동작을 선택할 수 있어요. 하지만 무엇을 원하는지 알아야 합니다.

— 고객사의 품질 책임자가 실제로 이 상황에서 무엇을 하는지도 물어봐야 해요. Lucas가 덧붙였습니다. 가설은 있지만 모두에게 맞는 규칙은 없어요.

Alice가 말했습니다.

— 기대하는 제품 동작을 명확히 할게요. 혼자 하지는 않을 겁니다. 결과를 확인하려면 지원팀과 Lucas, Thomas, 아마 고객 한 곳도 필요해요.

Vivien이 고개를 들었습니다.

— 그럼 내일 회의를 잡을게요.

Alice는 망설였습니다.

— 적절한 사람들을 모으는 건 도와줄 수 있어요. 하지만 우리 대신 결정을 내리는 건 Vivien의 몫이 아니에요.

그를 배제하려는 말은 아니었습니다. 그래도 Vivien은 자기 자리가 옮겨지는 듯한 느낌을 받았습니다.

수년 동안 그는 문제를 올바른 절차로 되돌려 보내는 사람이었습니다. 이번에는 관련자들이 이미 서로 이야기해야 한다는 사실을 알고 있었습니다. 의견 차이를 일련의 티켓으로 바꾸어 줄 필요도 없었습니다.

필요한 것은 다른 것이었습니다. 두 개발자 중 어느 쪽도 혼자 내릴 수 없는 판단이었습니다.

다음 날 Alice는 각자가 피하려는 것이 무엇인지 묻는 것으로 시작했습니다.

Lucas는 정보가 빠졌는데도 품질 책임자가 완전한 추적 정보를 가지고 있다고 믿게 되는 일을 피하고 싶었습니다.

Thomas는 모호한 규칙 때문에 과거 데이터가 불완전한 고객의 운영이 불필요하게 멈추는 일을 피하고 싶었습니다.

지원팀은 공급업체 경보가 언제나 즉각적인 리콜을 뜻하지는 않는다고 상기시켰습니다. 일부 고객은 먼저 관련 범위를 파악하고 품질팀과 논의한 뒤 취할 조치를 결정해야 했습니다.

— 좋아요. Alice가 말했습니다. 그렇다면 하나의 상태에 서로 다른 두 의미를 담으려 했던 게 실수일지도 모르겠네요.

Alice는 제안을 공유했습니다.

로트는 검토 필요 상태가 될 수 있습니다. 그렇다고 ERP가 고객사를 대신해 규제나 품질에 관한 결정을 내리는 척해서는 안 됩니다. 하지만 그 로트를 사용하거나 출하할 수 있는 흐름에서는 경보를 보이게 하고, 경보가 사라지게 두는 대신 정해진 확인 절차를 거치게 해야 합니다.

불확실한 과거 정보는 계속 불확실하다고 표시해야 합니다. 소프트웨어에 필요한 데이터가 없는 사례를 위험이 없는 것처럼 보여서도 안 됩니다.

— 이건 출하 모듈에서 확인할 수 있어요. Thomas가 말했습니다. 하지만 오래된 주문도 시험하고 싶습니다.

— 고객과 용어도 다시 검토하고 싶어요. Lucas가 덧붙였습니다. '검토 필요'가 그 품질 책임자에게는 같은 의미가 아닐 수도 있어요.

Alice는 동의했습니다.

설계가 끝난 것은 아니었습니다. 누가 무엇을 선택해야 하는지는 마침내 알아냈습니다.

Alice는 임시 결정과 열린 질문, 그 이유, 확인할 사례를 공유 컨텍스트에 적었습니다. 기준표에 조건도 추가했습니다. 국지적인 결정이 여러 모듈에서 쓰는 정보의 의미를 바꾼다면 코드가 되돌릴 수 있어 보여도 자동으로 초록에 해당하지 않습니다.

— 더 일찍 알았어야 했어요. Lucas가 말했습니다.

— 맞아. Alice가 대답했습니다. 하지만 그래서 테스트하는 거야. 다음에는 무엇을 봐야 할지 알게 됐잖아.

Thomas는 두 문서를 살펴보았습니다.

— 모듈을 가로지르는 테스트도 필요해요. 각 동작을 따로 확인하는 테스트는 많지만, 서로 만났을 때 어떤 일이 생기는지 확인하는 테스트는 훨씬 적어요.

몇 분째 듣고 있던 Vivien은 메모했습니다.

통화가 시작된 뒤 처음으로 익숙한 문제를 발견했습니다. 모든 의존 관계를 완전히 보는 사람이 아무도 없었습니다.

하지만 해결책이 만들어지는 방식은 인상적이었습니다. Alice는 화면을 지시하지 않았습니다. Lucas와 Thomas는 작업 분할을 기다리지 않았습니다. 지원팀은 고객의 실제 경험을 가져왔습니다. Vivien의 역할은 빠진 부분을 보이게 하고 대화를 돕는 것이었습니다.

유용한 일이었습니다.

매주 월요일 자신이 주관하던 출시 회의와는 거의 관련이 없었습니다.

오후에 Thomas는 다른 팀의 경력이 짧은 개발자에게 메시지를 받았습니다.

그 개발자는 오래된 재고 처리 로직이 왜 겉보기에는 같은 검사를 두 번 하는지 이해하지 못했습니다. 코딩 에이전트는 하나를 없애자고 제안했습니다.

Thomas는 파일을 열었습니다.

— 함께 살펴보기 전에는 건드리지 마세요. 그가 답했습니다.

두 사람은 30분 동안 화면을 공유했습니다. 두 번째 검사는 과거 제품 변경에서 물려받은 사례를 다루고 있었지만 코드에는 제대로 설명되어 있지 않았습니다. Thomas도 정확히 어느 출시에서 생겼는지는 기억하지 못했지만 어디를 찾아봐야 하는지는 알았습니다.

— 도구가 나쁜 답을 지어낸 건 아니에요. Thomas가 설명했습니다. 비슷한 검사 두 개를 보고 단순화하자고 제안한 거죠. 소프트웨어의 역사를 모른다면 왜 그렇게 판단했는지 이해됩니다.

— 정말 AI로 코딩하는 일이 전혀 없으세요? 개발자가 물었습니다.

— 이런 주제에서 아무 맥락 없이 수정안을 제안하게 하지는 않아요.

— 하지만 방금 30분을 썼잖아요. Lucas였다면 5분 만에 확인했을지도 모르는데요.

Thomas는 생각했습니다.

— 그럴지도 모르죠. Lucas도 같은 사실을 찾았을 수 있고요. 중요한 건 이 검사가 왜 존재하는지 이해하는 겁니다. 다음번에는 도구에도, 제게도 더 나은 질문을 할 수 있을 테니까요.

개발자는 설명을 문서화해도 되는지 물었습니다.

— 꼭 그렇게 해 주세요. 저 혼자 기억하는 건 이제 지쳤어요.

통화를 마친 Thomas는 자기가 한 마지막 말에 스스로 놀랐습니다.

수년 동안 예외에 관한 지식은 그를 없어서는 안 될 사람으로 만들었습니다. AI가 이 경험의 가치를 낮출까 두려울 때도 있었습니다. 하지만 이번 대화는 다른 가능성을 보여 주었습니다. 모든 답을 혼자 간직하는 대신 다른 사람들이 부족한 판단력을 얻도록 도울 수 있었습니다.

그는 공유 문서를 열고 몇 줄을 적었습니다.

월요일 미팅에서 Hubert는 Alice에게 실험의 진행 상황을 물었습니다.

Alice는 서로 양립하지 않는 두 결정과 그것을 드러낸 테스트, 틀을 수정한 일을 설명했습니다.

— 그렇다면 두 개발자가 결정하게 두었다가 거의 불일치가 생길 뻔한 거군요? Hubert가 요약했습니다.

— 네.

— 별로 안심이 되지는 않네요.

— 운영 환경에 배포하기 전에 발견했어요. Alice가 대답했습니다. 문제는 두 사람이 결정을 내렸다는 게 아니에요. 그들의 선택이 만나는 방식을 우리 규칙 어디에도 적지 않았다는 거예요.

Hubert는 사고 기록을 살펴보았습니다.

— 전 같았으면 검토 회의를 하나 더 했겠죠.

— 그랬을지도 몰라요. 하지만 회의를 하나 더 한다고 이 질문이 반드시 나타났을지는 몰라요. 전체 흐름을 시험하는 사람과, 선택이 여러 영역에 영향을 줄 때의 명확한 책임이 필요했어요.

— 이제는 누가 결정하죠?

— Lucas는 자기 주제를 맡아요. Thomas는 자기 주제를 맡고 기술 위험을 도와줘요. 저는 프로덕트 일관성의 보증인이에요. 민감한 사례에서는 관련 전문가에게 묻고요. 이 모든 걸 도구에 넘길 수는 없어요.

Hubert는 천천히 고개를 끄덕였습니다.

— Vivien은요?

Alice는 몇 초 뒤 답했습니다.

— 두 팀을 연결하고 어디서 막혔는지 이해하는 데 도움을 줬어요. 하지만 작업을 배분하거나 제품 동작을 결정할 필요는 없었어요.

Hubert는 다른 생각에 잠긴 듯했습니다.

— 금요일에 만났어요. 새 프로젝트 추적 자료를 보여 줬죠. 보고서는 아주 좋았습니다. 하지만 그가 말한 내용의 80%는 기계가 준비할 수 있을 것 같아요.

— Vivien도 알고 있어요. Alice가 대답했습니다.

— 그 사실에서 어떤 결론을 내려야 할지는 모르겠군요.

Alice에게도 준비된 답은 없었습니다.

— 그가 더는 쓸모없다는 결론은 아니에요. Alice가 마침내 말했습니다. 방금도 사람들을 함께 일하게 하는 법을 안다는 걸 봤잖아요. 다만 모든 것이 그를 거치는 직책일 필요는 더 이상 없을 뿐이에요.

Hubert는 화면을 계속 바라보았습니다.

— 그와 이야기해야겠군요.

Vivien은 다음 날 일정 초대를 받았습니다.

처음에는 조직 개편을 떠올렸습니다. 몇 주 전부터 일정에 빈 시간이 늘었습니다. 개발자들은 자신의 주제를 직접 관리했습니다. 도구가 Vivien이 전에는 저녁마다 모으던 정보의 일부를 자동으로 수집했습니다. 여전히 사람 사이의 어려움을 다루고, 신입을 적응시키고, 복잡한 상황을 풀었지만 그런 일만으로는 예전처럼 일정이 가득 차지 않았습니다.

Vivien은 Malorean의 출시 과정에 꼭 필요한 사람이라는 사실을 좋아했습니다.

이제 덜 필요한 사람이 되었다는 사실까지 좋아할지는 확신하지 못했습니다.

하지만 회의는 뜻밖의 질문으로 시작되었습니다.

— 앞으로도 어떤 일을 계속하고 싶나요, Vivien? Hubert가 물었습니다.

— 제 일이요. 그가 대답했습니다.

Hubert는 미소를 지었습니다.

— 바로 그 일이 무엇인지 이해하려는 겁니다.

Vivien은 팀의 운영 방식을 설명하려고 초대된 Alice를 바라본 뒤 Hubert에게 시선을 돌렸습니다.

— 모르는 척하고 싶지는 않습니다. 팀은 이제 제가 매주 어디까지 했느냐고 물을 필요가 없어요. 에이전트가 이미 준비한 요약을 읽으며 시간을 보내고 싶지도 않고요.

— 아직 팀에 부족한 것은 무엇인가요? Alice가 물었습니다.

— 어디에도 기록되지 않은 갈등. 길을 잃었다는 말을 망설이는 사람들. 누구에게 물어야 할지 모르는 신입. 여러 팀을 가로지르는 결정. 그리고 때로는 그저 시간을 내어 들어 줄 사람.

그는 잠시 멈췄습니다.

— 그렇다고 이 모든 일이 계속 저를 거쳐야 한다는 뜻은 아닙니다.

Hubert는 자세를 바로잡았습니다.

— 그렇죠. 역할이 바뀐다면 누가 이 일을 책임지는지 분명히 해야 합니다. 직책을 하나 없앤 뒤 아무도 사람들을 돌보지 않는다는 사실을 발견하고 싶지는 않아요.

Vivien은 동의했습니다.

세 사람은 개발자 지원과 팀을 가로지르는 판단, 경영진이 계속 추적해야 할 것을 논의했습니다. Alice는 Thomas가 동료들에게 오래된 코드를 더 잘 이해하도록 돕는 역할을 이야기했습니다.

통화에서는 최종적인 해결책이 나오지 않았습니다.

회의가 끝난 뒤 Vivien은 잠시 일정을 바라보았습니다.

전날보다 덜 위협받는 느낌이었지만 그렇다고 안심되지는 않았습니다. 그의 직업은 사라진 것이 아니라 여러 책임으로 분해되고 있었고, 그중 일부는 다른 방식으로 맡을 수 있었습니다.

그날 저녁 Vivien은 평소보다 일찍 컴퓨터를 닫았습니다.

오랜만에 따라잡아야 할 출시 지연이 없었습니다.

그리고 되찾은 이 시간을 무엇에 쓸지는 아직 알지 못했습니다.

10 – AI는 기업의 중간 계층을 사라지게 할까?

오늘날 팀은 의사결정을 직접 찾아보고, 알려진 틀 안에서 행동하며, 진행 상황 추적의 일부를 에이전트에게 맡길 수 있습니다. 정보를 전달하거나 업무의 모든 단계를 조직해 줄 책임자가 필요한 경우가 줄어듭니다.

그렇다면 그 역할을 맡았던 중간 관리 계층은 어떻게 될까요?

답은 분명해 보입니다. AI가 조율을 맡으면 중간 관리자는 쓸모를 잃는다는 것입니다. 그러나 관리자는 지시를 전달하는 일만 하지 않습니다. 판단하고, 지원하고, 팀을 연결하며, 어떤 절차에도 들어맞지 않는 문제를 맡습니다.

따라서 진짜 질문은 얼마나 많은 직책이 사라질지가 아니라 그 기능과 권력이 어디로 갈 것인지 이해하는 데 있습니다.

의사결정을 조율하기 위해 만들어진 위계

중요한 고객이 문제를 신고합니다. 지원팀은 책임자에게 전달하고, 책임자는 프로덕트 팀에 연락합니다. 프로덕트 팀은 엔지니어링에 추정을 요청합니다. 여러 책임자가 정보를 모아 긴급성을 비교한 뒤 필요하면 판단을 한 단계 더 위로 올립니다.

이 흐름이 반드시 불합리한 것은 아닙니다. 각 책임자는 맥락의 일부를 가지고 있고 회사의 일부를 움직일 권한이 있습니다. 이 연쇄 구조가 문제를 이해하고 결정하는 유일한 방법이 될 때 부담스러워집니다.

역사적으로 위계는 정보가 분산된 상태에서도 기업이 성장할 수 있게 했습니다. 중간 계층은 경영진의 목표를 우선순위로 번역하고, 팀을 조율하고, 의존 관계를 살피며, 자원 충돌을 판단합니다. 직원 평가와 성장, 사람 사이의 어려움을 해결하는 일에도 참여합니다.

하지만 이 모든 활동의 성격이 같은 것은 아닙니다. 장애 보고서를 준비하는 일과 사업 기회를 포기하기로 결정하는 일은 다릅니다. 마감일을 추적

하는 일에는 갈등을 해결하거나 누군가의 성장을 돕는 일과 같은 판단이 필요하지 않습니다.

저는 Malorean에서 프로덕트 오너를 두고 비슷한 구분을 주장했습니다. 단순한 전달자 역할을 하는 직책은 존재 이유를 잃더라도 본래 맡아야 했던 기능까지 불필요해지는 것은 아닙니다. 그 일을 맡은 사람의 직함만이 아니라 실제 수행되는 일을 물어야 합니다. [1]

중간 계층이 줄어든다면 그들이 맡았던 기능은 누가 수행할까요?

원격 근무는 이미 조율의 규칙을 바꾸었다

이 질문을 만든 것은 AI가 아닙니다. 여러 국가와 시간대에서 원격으로 일하는 기업들은 복도 대화나 즉석 회의에 기대지 않고 결정하는 법을 배워야 했습니다.

GitLab의 'handbook-first' 원칙은 규칙과 맥락에 글로 접근할 수 있게 합니다. 누구나 수정을 제안할 수 있고 해당 영역의 책임자가 반영 여부를 결정합니다. 직원은 관리자가 전달할 때까지 기다리지 않고 정보를 찾을 수 있습니다. [2]

Alan은 다르지만 상호 보완적인 관행을 소개합니다. 중요한 결정은 GitHub 스레드에서 논의합니다. 한 사람이 결정하고 실행하며, 다른 한 사람이 주제의 성공을 계속 책임지고, 관련자는 자문을 받거나 정보를 전달받습니다. 만장일치를 요구하지 않고 토론을 열 수 있습니다. [3]

Basecamp를 만든 37signals에서는 상당한 조율이 글로 된 대화와 보고, 일시적인 대화 속에 흩어지지 않고 프로젝트와 함께 보존되는 정보를 통해 이루어집니다. [4]

이 세 회사에 관리자가 없다는 뜻은 아닙니다. 정보를 쥐고 있거나 결정된 사항을 계속 상기시켜야 하는 한 사람에게 의존하는 일을 줄이는 세 가지 방식입니다.

이 선례는 AI 에이전트에게 중요합니다. 조직이 정보를 명시했다면 시스템도 규칙을 확인하고, 이전 판단을 찾고, 책임자를 식별할 수 있습니다.

관리자의 머릿속에만 남은 결정은 동료에게도 에이전트에게도 접근할 수 없습니다.

다만 원격 근무를 이상화해서는 안 됩니다. Microsoft가 직원 6만 1천여 명을 대상으로 한 연구에서는 원격 근무 전환 뒤 협업 네트워크가 더 고립되는 현상도 관찰했습니다. 더 많이 쓴다고 맥락 공유나 팀 간 협력이 보장되지는 않습니다. [5]

원격 근무가 남긴 진짜 유산은 사무실의 소멸이 아니라, 상시 중개자 없이 접근할 수 있는 정보와 결정을 중심으로 업무 일부를 조직할 가능성입니다. AI는 이 성과를 활용할 수 있지만 혼자 만들어 내는 것은 아닙니다.

AI가 이 업무의 일부를 맡기 시작한다

문서는 규칙을 보존하고 대시보드는 지연을 보여 줍니다. 에이전트는 한 걸음 더 나아가 여러 정보를 연결하고, 진행 상황을 준비하고, 모순을 감지하고, 처리 방안을 제안하거나 허용된 행동을 실행할 수 있습니다.

이는 관리자가 시간의 일부를 쓰던 활동과 정확히 겹칩니다. 통합, 보고, 추적, 판단 준비입니다. 비용은 줄어들 수 있지만 활동 자체가 사라진다고 단정할 수는 없습니다.

45건의 인터뷰 등을 바탕으로 한 HEC Paris의 탐색적 분석은 구조화되고 데이터가 풍부한 관리 업무가 신뢰와 인간관계, 판단에 의존하는 업무보다 자동화에 더 적합하다고 봅니다. 실제로 사라진 관리 시간이나 불필요해진 직책을 측정한 연구는 아닙니다. [6]

Procter & Gamble에서 진행한 실험은 프로덕트 혁신 상황에서 AI와 함께 일한 한 사람이 AI 없이 일한 두 사람의 팀과 비슷한 결과를 얻을 수 있음을 보여 줍니다. 도구는 특히 서로 다른 직군의 관점을 연결하는 데 도움을 줍니다. 하지만 일시적인 협업을 쉽게 하는 것이 장기적으로 책임자를 대체한다는 뜻은 아닙니다. [7]

66개 기업의 직원 7,137명을 대상으로 한 다른 실험에서는 AI 도우미를 정기적으로 사용한 사람들이 연구 후반기에 매주 약 두 시간의 이메일 시간

을 절약했습니다. 그러나 연구진은 6개월 동안 업무 구성의 유의미한 변화를 관찰하지 못했습니다. 이 연구는 전체 프로세스를 혼자 실행하는 에이전트가 아니라 사무용 도우미를 다룹니다. [8]

이 간극은 결정적입니다. 개인의 생산성 향상이 곧 조직 변화는 아닙니다.

한 계층의 필요성이 줄어들려면 정보가 실제로 팀에 도달하고, 의사결정 권한이 바뀌며, 이례적인 상황을 판단할 곳이 있어야 합니다. 이는 도구만의 문제가 아니라 조직의 문제입니다.

서로 매우 다른 세 기업의 경로

모든 조직이 같은 방식으로 반응하지는 않습니다. 세 가지 사례를 통해 이미 시작된 조직 개편, AI를 중심으로 세운 포부, 급격한 변화의 어려움을 구분할 수 있습니다.

Bayer: 실제로 줄어든 위계

Bayer는 *Dynamic Shared Ownership*(DSO) 모델을 통해 의사결정 일부를 다직군 팀에 재분배했습니다. 2025년 주주 서한에서 그룹 대표는 조직 계층을 최대 여섯 단계 없애고 관리직을 약 3분의 2 줄였다고 밝힙니다. 의사결정이 빨라지고 개발 기간이 단축되었다고 주장합니다. [9]

여기서 효과를 보고한 주체가 Bayer라는 점을 감안해도 규모는 주목할 만합니다. 이런 변화에는 조직도의 직함만이 아니라 일상의 책임까지 바꿔야 합니다. 팀은 현장에서 더 많이 결정하고, 책임자는 지원에 더 많은 시간을 쓰며, 일부 조율 기능은 재편됩니다.

다만 사례를 다르게 읽게 하는 중요한 사실이 있습니다. **DSO**는 생성형 **AI** 에이전트가 부상하기 전에 시작되었습니다**.** Bayer는 대기업도 위계를 크게 줄일 수 있음을 보여 주지만 AI가 그 원인이라는 증거는 아닙니다.

Block: 위계 전체를 거치지 않고 컨텍스트 전달하기

Block은 AI에 부여하려는 위치에서 더 나아갑니다. Jack Dorsey와 투자자 Roelof Botha가 공동 집필한 글에서 경영진은 공유 정보 시스템을 기반으로 한 조직을 설명합니다.

직원이 책임자의 맥락 전달을 기다리지 않고 상황을 이해해 행동할 수 있게 하려는 것입니다. 저자들은 기여자, 여러 영역에 걸친 결과를 맡는 사람(DRI), 직접 생산하면서 다른 사람도 지원하는 책임자인 *player-coach*를 구분합니다. [10]

주로 전달자 역할을 하던 관리자의 중요성은 줄어들 것입니다. 반면 사람의 성장과 결과에 대한 책임, 방향을 결정하는 일은 남을 것입니다.

이는 회사와 투자자가 설명한 포부이지, 모델의 장기적인 성공을 증명한 독립적인 평가가 아닙니다.

Meta: 생산량이 통제 역량을 넘어설 때

반대 사례는 2026년 Meta의 조직 개편 계획을 다룬 Reuters 조사에서 찾을 수 있습니다. 더 작은 팀, AI의 큰 역할, 특정 계층의 축소가 계획되거나 실행되었습니다. 결국 두 번째 조직 개편은 취소되었습니다.

Reuters가 인용한 내부 문서에 따르면 내부 소프트웨어와 인프라 변경은 사용자에게 출시된 새 기능보다 훨씬 빠르게 늘었습니다. 동시에 기술 장애와 해결에 쓰는 시간도 증가했습니다. Meta는 보도의 일부 내용을 부인하거나 다른 해석을 제시했습니다. [11]

여러 변화가 동시에 일어났으므로 이 수치만으로 장애를 AI나 관리자 축소 탓으로 돌릴 수는 없습니다. 하지만 결정적인 질문을 제기합니다. 회사가 생산한 것을 검증하고 안정시키며 흡수하지 못한다면 더 많이 만드는 것이 무슨 의미가 있을까요?

Bayer는 조직을 평평하게 만들 가능성을 보여 주고, **Block**은 **AI**를 염두에 둔 형태를 그리며, **Meta**는 생산량이 조율보다 빨리 늘어나는 조직 개편의 위험을 상기시킵니다. 어느 경로도 혼자서 경영의 미래를 예고하지는 않습니다.

8

담당자가 사라져도 관리 기능은 사라지지 않는다

예를 들어 고객이 소프트웨어 오작동을 신고했다고 합시다. 알려진 문제이고 해결 규칙이 분명하다면 팀은 여러 승인 단계를 기다리지 않고 행동

할 수 있습니다. 신뢰할 수 있는 데이터로 조사하고 수정한 뒤 다른 사람에게 알릴 수 있습니다.

그런데 두 진단이 서로 모순됩니다. 에이전트는 한 행동을, 개발자는 다른 행동을 권합니다. 수정 사항은 인접 서비스에 문제를 일으킬 수 있습니다.

이때는 누군가 의견 차이를 드러내고, 필요한 역량을 모으고, 어떤 위험을 감수할지 결정해야 합니다. 자율성이 조율을 없앤 것이 아니라 수행 방식을 바꾼 것입니다.

Martin Krzywdzinski와 공동 연구자들의 실험은 흥미로운 단서를 제공합니다. 모의 생산 라인에서 87개의 2인 팀이 때로는 틀리는 AI 시스템의 도움을 받아 고장에 대응했습니다. 자율적으로 조직된 팀은 지정된 리더가 있는 팀보다 더 많이 소통했고 문제도 더 잘 해결했습니다. 분석은 소통의 결정적인 역할을 강조합니다. [12]

실험실의 두 사람이 대기업을 대표하지는 않습니다. 그래도 연구는 중요한 구분을 상기시킵니다. 팀은 공식적인 리더 없이 일할 수 있지만 조율이라는 업무 없이 일할 수는 없습니다.

동료나 전문가, 결정을 위임받은 사람, 직속 관리자가 그 일을 맡을 수 있습니다. 필요한 역량과 위험, 의견 차이에 따라 선택은 달라집니다.

자동 보고만으로 충분히 해낼 수 없는 일도 남습니다. 사람을 맞이하고, 성장시키고, 긴장을 풀고, 어려운 결정을 지원하고, 성과를 인정하는 일입니다. 모든 관리자가 이런 일을 잘하는 것도 아니며 다른 방식으로 나눌 수도 있습니다. 그래도 누군가는 그 임무와 시간을 가져야 합니다.

Gallup은 AI 사용이 고용 불안을 없애지 않으며 직장에서 지원받는다고 느끼는 사람일수록 불안이 낮은 경향이 있다고 관찰합니다. 이것이 전통적인 관리자의 필요성을 증명하지는 않지만 조직이 인간 공동체이기도 하다는 사실을 상기시킵니다. [13]

그리고 미래에 어려운 결정을 내려야 할 사람은 누가 훈련할까요?

저는 Malorean에서 주니어 개발자에게 생길 수 있는 위험을 언급했습니다. AI가 즉시 코드를 작성하고 오류를 수정하면 디버깅하고 실패를 이해하며 판단력을 쌓을 기회가 사라질 수 있습니다. 연구해야 할 우려이지 역량이 전반적으로 하락했다는 증거는 아닙니다. [14]

미래의 책임자도 같은 문제를 만납니다. 판단을 지켜보고, 피드백을 받고, 팀이 갈등에서 빠져나오는 과정을 보면서 배우기도 합니다. 계층을 없애는 기업은 이 경험을 전달할 다른 길을 마련해야 합니다.

결국 각 기능에는 여러 목적지가 있을 수 있습니다. 자동화하거나, 팀이 맡거나, 전문가에게 넘기거나, 경영진에게 올릴 수 있습니다. 하나의 작업은 사라지거나, 다른 사람에게 넘어가거나, 누군가의 업무에 추가될 수 있습니다.

따라서 더 평평한 조직의 효과를 판단할 때 없앤 직책만 세어서는 안 됩니다. 사라진 추적 업무와 새롭게 생긴 에이전트 감독, 장애 해결, 지원, 교육 부담을 비교해야 합니다. 이 재분배의 견고한 순효과는 아직 알지 못합니다.

더 평평한 조직은 정말 더 자율적인가?

조직도에서 잘 보이지 않는 마지막 질문이 남습니다. 중개자가 사라질 때 누가 결정할까요?

정보를 더 잘 알고 더 많은 문제를 위로 올리지 않고 해결하는 자유로운 팀을 상상할 수 있습니다. 하지만 반대도 가능합니다. 현장 책임자는 줄어들고 경영진이 시스템을 통해 더 많은 결정을 직접 통제할 수 있습니다.

이 긴장은 생성형 AI 이전부터 존재했습니다. 산업 기업을 연구한 Nicholas Bloom과 공동 연구자들은 직원에게 더 많은 정보를 주는 도구가 현장의 자율성 증가와 연관된다고 봅니다. 반대로 상사와의 소통을 쉽게 하는 도구는 중앙집중화를 촉진할 수 있습니다. 위에 지시를 묻기가 더 쉬워지기 때문입니다. [15]

이 선례가 현재 에이전트에 관해 무언가를 증명하지는 않습니다. 다만 한 메커니즘을 밝혀 줍니다. 데이터를 널리 나누는 것과 권력을 분산하는 것은 같은 일이 아닙니다.

알고리즘 관리 시스템은 이 위험을 더 구체적으로 만듭니다. 업무를 배정하고, 우선순위를 제안하고, 성과를 측정하고, 경보를 발생시킬 수 있습니다. 연구에 따르면 인간 관리자가 적용이나 예외 처리를 위해 남아 있어도 일부 지휘·평가 기능은 이런 도구가 맡을 수 있습니다. [16]

시스템은 조직도에서 상사처럼 보이지 않습니다. 하지만 각자에게 어떤 업무를 줄지, 어떤 기준으로 결과를 평가할지, 어떤 조건에서 행동을 허용할지를 결정한다면 권한의 일부를 구조화합니다.

그러므로 화면 뒤의 결정을 보아야 합니다. 누가 에이전트의 목표를 정하는가? 누가 규칙을 선택하고 바꿀 수 있는가? 누가 이의를 제기할 수 있는가? 알고리즘이 틀렸을 때 실제로 그 결과를 취소할 권한은 누구에게 있는가?

팀이 막대한 실행 역량을 가지고도 우선순위를 바꿀 수 없을 수 있습니다. 다른 팀은 관리자를 두고도 수단과 현장의 판단을 실제로 스스로 결정할 수 있습니다.

저는 Malorean에서 선택과 받아들인 손실, 결정자를 명시하지 않은 채 합의처럼 보이게 하는 '어정쩡한 결정'을 이미 설명했습니다. 권력이 여전히 식별하기 어렵다면 계층 하나를 없애도 달라지는 것은 없습니다. [17]

따라서 눈에 보이는 위계는 줄어들지만 도구 안의 중앙 통제는 강화되는 역설을 상상할 수 있습니다.

필연은 아닙니다. 같은 시스템이 팀의 현장 결정을 돕고 권력의 규칙을 더 투명하게 만들 수도 있습니다. 중요한 것은 조직도에서 지운 칸의 수가 아니라 회사가 실제로 무엇을 위임하기로 했는가입니다.

실제로 사라질 수 있는 것

AI는 기업의 중간 계층을 사라지게 할까요?

일부 계층의 필요성은 줄어들 수 있습니다. 하지만 자동으로 사라지는 것도, 사라진다고 경영이 끝나는 것도 아닙니다.

주로 정보를 전달하고, 진행 상황을 통합하고, 반복적인 검토를 조직하는 기능은 비중이 줄어들 수 있습니다. 판단과 갈등, 경험 전수, 책임으로 이루어진 다른 기능은 다른 방식으로 맡아야 합니다.

Bayer는 AI가 확인된 원인이 아니어도 계층을 대대적으로 줄일 수 있음을 보여 줍니다. Block은 조율을 공유 시스템에 더 많이 맡길 것을 제안합니다. Meta는 신뢰성과 인간의 부담을 측정하지 않고 실행 속도를 높이면 조직이 약해질 수 있음을 상기시킵니다. 에이전트 때문에 경영이 전반적으로 사라진다는 증거는 아직 없습니다.

그럼에도 이동은 깊을 수 있습니다. 책임자는 정보나 승인을 얻기 위해 반드시 거쳐야 하는 관문 역할을 덜 하게 될 것입니다. 대신 일관성과 예외 상황, 사람의 성장에 더 많은 주의를 기울일 수 있습니다. 어떤 곳에서는 이런 기능의 일부를 팀이 직접 맡고, 다른 기능은 소프트웨어에 새겨지거나 소수 의사결정자에게 집중될 것입니다.

불필요해지는 것은 반드시 관리 자체가 아닙니다. 다시 검토되기 시작한 것은 필수 조율 메커니즘으로서의 위계입니다.

개발자에게 그 결과는 구체적입니다. 업무가 더 적은 중개자를 거칠 때 생산 수단만 더 많이 받는 것이 아닙니다. 필요와 절충안, 책임져야 할 결과를 더 직접 마주합니다.

이것이 첫 번째 책 전체가 설명한 움직임입니다. 행동할 역량이 커지고, 그와 함께 이해하고 선택하고 책임질 필요도 커집니다.

이 변화가 더 나은 결정이나 더 인간적인 조직을 보장하지는 않습니다. 다만 조직을 다르게 설계하도록 요구합니다. 컨텍스트를 공유하고, 권력을 식별할 수 있게 하고, 경험 전수를 보존하고, 결과에 책임져야 합니다.

중요한 것은 AI가 이제 무엇을 생산할 수 있게 하는지만이 아니라, 이 새로운 역량을 중심으로 어떤 회사를 만들 것인가입니다.

마지막 이야기에서 Malorean Corp.로 돌아가 경영진과 팀이 조직도를 넘어 자신들이 내린 선택을 어떻게 살아가는지 살펴보겠습니다.

참고문헌

[1] Le PO n'est pas un métier, c'est une fonction – et l'IA n'y change rien, elle accélère – Malorean

- URL: https://malorean.net/articles/2026-08-03-le-po-nest-pas-un-metier-cest-une-fonction.html
- 결정과 실행 사이의 전달자 역할을 주로 하는 직책의 존재와 한 기능이 수행하는 유용한 업무를 구분한 저자의 글입니다.

[2] Handbook First – GitLab

- URL: https://handbook.gitlab.com/handbook/company/culture/all-remote/handbook-first/
- 문서를 공유 기준으로 삼고 수정 사항을 공개적으로 논의하면서 인간 책임자가 결정을 계속 맡는 방식을 소개합니다.

[3] Comment prendre des décisions dans un environnement de travail asynchrone – Alan

- URL: https://alan.com/fr-fr/blog/healthy-business/a/decisions-asynchrones-methode-alan
- 결정을 글로 논의하는 방법과 누가 결정하고 책임지며 누구에게 자문하거나 알려야 하는지를 설명합니다.

[4] The 37signals Guide to Internal Communication – 37signals

- URL: https://37signals.com/how-we-communicate
- 글로 된 대화와 시간이 지나도 유지되는 컨텍스트를 활용하는, 주로 비동기적인 사내 소통 방식을 설명합니다.

[5] The effects of remote work on collaboration among information workers – Microsoft Research

- URL: https://www.microsoft.com/en-us/research/publication/the-effects-of-remote-work-on-collaboration-among-information-workers/
- 원격 근무 전환 뒤 협업 네트워크를 연구하고 고립이 심해질 위험을 보여 줍니다.

[6] AI Automates Routine Management and Raises the Value of Leadership – HEC Paris

- URL: https://www.hec.edu/en/dare/tech-ai/ai-automates-routine-management-and-raises-value-leadership
- 인터뷰와 모델 평가를 바탕으로 관리 업무를 탐색적으로 분석하며 직책 감소의 인과적 측정은 제시하지 않습니다.

[7] The Cybernetic Teammate: A Field Experiment on Generative AI Reshaping Teamwork and Expertise – NBER

- URL: https://www.nber.org/papers/w33641
- Procter & Gamble에서 AI가 혁신 업무와 서로 다른 전문성의 결합에 기여하는 방식을 살펴본 실험입니다.

[8] Shifting Work Patterns with Generative AI – NBER

- URL: https://www.nber.org/papers/w33795
- 66개 기업에서 진행한 무작위 실험으로, 개인의 시간 절약은 나타났지만 6개월 동안 집단 업무의 유의미한 변화는 관찰되지 않았습니다.

[9] Chairman's Letter – Bayer Annual Report 2025

- URL: https://reports.bayer.com/annual-report-2025/en/to-our-stockholders/chairmans-letter.html
- Bayer가 에이전트 이전에 시작한 DSO 변화에서 관리 계층과 직책을 크게 줄였다고 보고합니다.

[10] From Hierarchy to Intelligence – Sequoia Capital / Block

- URL: https://sequoiacap.com/article/from-hierarchy-to-intelligence
- Jack Dorsey와 Roelof Botha가 DRI와 player-coach를 포함한 공유 시스템 기반 조율의 비전을 제시합니다.

[11] Mark Zuckerberg had a bold plan to replace Meta staff with AI. Here's how it imploded – Reuters

- URL: https://www.timeslive.co.za/news/sci-tech/2026-08-26-special-report-mark-zuckerberg-had-a-bold-plan-to-replace-meta-staff-with-ai-heres-how-it-imploded/
- TimesLIVE에 재게재된 Reuters 조사로, Meta의 변화 계획에서 나타난 조직·기술적 긴장을 다룹니다.

[12] How team organization influences the ability to solve automation failures – AI & Society

- URL: https://link.springer.com/article/10.1007/s00146-025-02761-5
- 자동화 고장에 대응한 자율적·위계적 2인 팀을 비교하고 소통의 역할을 측정한 실험 연구입니다.

[13] Using AI More Does Not Reassure Workers, Managers Do – Gallup

- URL: https://www.gallup.com/workplace/713231/ai-not-reassure-workers-managers-do.aspx
- AI에 대한 직원의 불안과 존중받는다는 느낌, 직장에서 받는 지원을 조사합니다.

[14] L'IA ne devrait pas seulement servir à produire dix fois plus. Elle devrait aussi nous obliger à comprendre dix fois mieux ce que nous produisons – Malorean

- URL: https://malorean.net/articles/2026-07-23-lia-produire-dix-fois-plus-comprendre-dix-fois-mieux.html

- AI가 주니어 개발자를 성장시키던 작업과 수정을 대신할 때 사라질 수 있는 학습 기회를 고찰한 저자의 글입니다.

[15] The distinct effects of Information Technology and Communication Technology on firm organization – NBER

- URL: https://www.nber.org/papers/w14975
- 정보 접근과 통신 기술이 현장의 의사결정 자율성에 서로 반대되는 영향을 줄 수 있음을 구분합니다.

[16] Algorithmic Management: From Technology to Politics and Theory – Weizenbaum Journal of the Digital Society

- URL: https://ojs.weizenbaum-institut.de/index.php/wjds/article/view/4_3_9/163
- 알고리즘 시스템이 수행하는 지휘·평가 기능과 인간이 개입할 수 있는 역량을 종합합니다.

[17] Pourquoi les organisations préfèrent les décisions molles – Malorean

- URL: https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.html
- 판단을 늦추고 받아들인 선택을 가리며 책임자를 식별하기 어렵게 만드는 표면적 합의를 다룬 글입니다.

11 – Malorean Corp. – 되찾은 시간

휴가 모듈을 시연한 지 두 달 뒤, Hubert Félix Duca는 개발 진행 상황을 묻지 않고 월요일 회의를 시작했습니다.

— 지난주에는 무엇을 배웠나요?

Alice가 화면을 공유했습니다.

최근 고객 피드백과 진행 중인 실험 결과, 아직 결론 내리지 않은 결정 몇 가지를 정리한 문서가 앞에 있었습니다. 에이전트가 밤사이 정보를 먼저 묶어 놓았습니다. Alice는 아침에 내용을 다시 읽고 잘못 연결한 부분 몇 가지를 제외한 뒤, 에이전트 혼자서는 만들 수 없었을 질문 두 가지를 추가했습니다.

— 추적 기능에서는 데이터가 완전할 때 시범 고객이 관련 제조 건을 훨씬 빨리 찾을 수 있었어요. Alice가 말을 시작했습니다. 하지만 한계를 명확히 보여 줘야 하는 과거 사례가 여전히 있습니다. 품질 책임자와 새로운 표시 방식을 시험할 예정이에요.

Hubert는 고개를 끄덕였습니다.

— 다른 고객들은요?

— 결론을 내리기에는 너무 일러요. Alice가 대답했습니다. 그래서 아직 배포를 일반화하지 않은 거고요.

Bob이 이어서 말했습니다. 그는 다른 프로덕트 주제들을 맡고 있었고 사전 영업팀과 지원팀을 여러 번 만난 뒤였습니다.

— 겉보기에는 비슷한 요청도 세 가지 있어요. 에이전트는 하나의 기능 아이디어로 묶었지만 뒤에는 적어도 두 가지 다른 문제가 있습니다. 투자 여부를 결정하기 전에 타깃부터 분명히 하고 싶어요.

— 개발자가 몇 명 필요하죠? Hubert가 물었습니다.

Bob은 미소를 지었습니다.

— 지금은 한 명도 필요 없습니다.

Hubert도 미소를 보냈습니다.

Bob은 알림 하나를 확인한 뒤 덧붙였습니다.

— 그리고 지원팀이 어제부터 반복 요청을 새롭게 분류하는 방식을 시험하고 있어요. 팀원 한 명이 익명화한 사본에서 에이전트와 함께 준비했어요. 정보가 적절한 담당자에게 더 빨리 가는지 보려고요.

— 누가 시작했죠? Hubert가 물었습니다.

— 지원팀이요. Alice가 사람의 결과 검토를 포함해 내부에서만 사용하고 되돌릴 수 있는 범위라는 걸 확인했습니다.

— 당신도 알고 있었나요?

— 아이디어는요. 테스트의 세부 사항은 몰랐습니다.

Hubert는 왜 시작하기 전에 아무도 알리지 않았느냐고 묻고 싶었습니다. 하지만 참았습니다. 제한적이고 되돌릴 수 있는 시도에는 더 이상 그의 승인이 필요하지 않다고 바로 그들이 합의했습니다.

— 효과가 있는지는 알려 주세요. 그가 마침내 말했습니다. 누군가 유지해야 한다면 담당자가 누구인지도 알아야 하고요.

테스트를 승인할 필요는 없었습니다. 다만 일반화할 경우 투입할 자원은 여전히 그의 책임이었습니다.

얼마 전까지만 해도 프로덕트 회의에서 이런 대답은 이상하게 들렸을 것입니다. 회의는 주로 개발자가 무엇을 어떤 순서로 만들어야 하는지 정했습니다. 이제 Alice와 Bob은 해결할 문제를 고르고, 시도가 일관된 범위에 머무는지 확인하며, 결정이 정해진 틀을 넘어설 때 개입하는 데 더 많은 시간을 썼습니다.

직책은 달라지지 않았습니다. 행정 문서에서 두 사람은 여전히 프로덕트 매니저였습니다.

하지만 일상 업무는 두 달 전과 완전히 같지 않았습니다.

가장 눈에 띄는 변화는 아마 더 이상 특별한 사건이 되지 않는 일이었습니다.

개발자가 고객과 이야기했습니다. 다른 개발자는 자기 모듈의 개선을 위해 Mini PRD를 제안했습니다. 영업 담당자는 전용 환경에서 시연의 변형을 만들었습니다. 지원팀 직원은 프로덕트 규칙의 불일치를 직접 알렸습니다.

직군의 경계를 두고 더는 큰 논쟁이 일어나지 않았습니다.

Malorean은 모두가 모든 일을 할 수 있다고 발표하지 않았습니다. 단지 어떤 행동이 반드시 그 직책을 평소 맡는 사람을 거쳐야 한다고 가정하는 일을 멈췄습니다.

각 다섯 명으로 이루어진 두 개발팀은 그대로 존재했습니다. 팀원의 대다수는 이제 특정 주제를 처음부터 끝까지 맡았습니다. 문제를 이해하고, 범위를 제안하고, 관련자와 이야기하고, 개발하고, 테스트한 뒤 결과를 추적했습니다.

이 방식에 다른 사람보다 빨리 적응한 개발자도 있었습니다. 두 명은 여전히 범위를 정할 때 가까운 지원을 받는 편을 선호했습니다. 다른 한 명은 복잡한 기술 문제에 탁월했지만 고객 미팅을 늘리고 싶은 마음은 전혀 없었습니다. 누구도 그들을 억지로 Lucas의 복제본으로 만들려 하지 않았습니다.

티켓도 남아 있었습니다.

개입을 조율하고, 정확한 작업을 기억하고, 수정 사항을 추적하는 데 사용했습니다. 하지만 주제의 존재 이유와 과거의 모든 결정, 에이전트에게 필요한 정보를 티켓 하나에 모두 담으려 하지는 않았습니다.

대신 각 주제 책임자가 공용 문서 도구에 컨텍스트 공간을 유지했습니다. 목표와 결정, 출처, 위험, 열린 질문, 그리고 이미 존재하는 자료로 가는 링

크가 있었습니다. 티켓, 견적, 지원팀 피드백, 미팅 기록, 테스트, 때로는 다시 찾아낸 옛 논의였습니다.

모든 것을 복사할 필요는 없었습니다. 중요한 것은 누군가 결정의 이유와 정보를 확인할 곳, 그 정보를 바꿀 수 있는 사람을 이해할 수 있게 하는 것이었습니다.

Alice의 첫 문서가 본보기가 되었습니다.

Bob이 방식을 가져갔고, 뒤이어 개발자들과 지원·영업·배포팀의 여러 사람이 따랐습니다. Alice처럼 많은 사람이 혼자서는 며칠 걸려 형식화했을 선택을 ChatGPT에 한 시간 동안 말로 설명할 수 있다는 사실을 발견했습니다.

사람들의 머릿속에 있던 모든 것이 마법처럼 접근 가능해진 것은 아닙니다. 하지만 이유와 예외, 암묵지의 일부는 접근할 수 있게 되기 시작했습니다.

원격 근무는 이 움직임을 더 자연스럽게 만들었습니다. 코로나19 이후 Malorean은 프랑스 전역에 흩어진 동료들이 복도 대화를 기다리지 않고 일할 수 있도록 더 많이 기록하는 법을 이미 배웠습니다. AI는 지식의 추출과 정리를 가속했지만 공유할 필요를 발명하지는 않았습니다.

Alice는 *프로덕트 결정 — 공통 규칙*이라는 페이지를 열었습니다.

— 지난주에도 혼동이 있었어. Alice가 Bob에게 설명했습니다. 한 팀 문서에서는 '승인됨'을 '기술적으로 검토됨'이라는 뜻으로 썼고, 지원팀은 '고객과 동작을 확인함'이라는 뜻으로 썼어.

Bob은 한숨을 쉬었습니다.

— 그 말은 누구에게나 같은 뜻이었던 적이 없지.

— 맞아. 로트 사건 이후 내린 결정으로 가는 링크를 추가했어.

몇 주 전 Lucas와 Thomas는 추적 기능과 출하 사이에 서로 양립하지 않는 두 동작을 거의 남길 뻔했습니다. 변경 사항은 테스트 중에 중단되었습니다. 그 뒤로 팀은 한 모듈 안에서는 자명해 보여도 다른 곳에서는 의미가 바뀌는 용어를 경계하게 되었습니다.

컨텍스트는 도움이 되었습니다. 새로운 문제도 만들었습니다.

일부 페이지는 최신 상태가 아니었습니다. 임시 결정을 확정 규칙처럼 복사한 경우도 있었습니다. 한 에이전트는 과거의 영업 가설을 회사의 약속인 것처럼 다시 꺼내기도 했습니다.

규율을 도입해야 했습니다. 각 컨텍스트에는 책임자와 출처, 확정된 결정, 아직 불확실한 항목을 밝혔습니다. 구조적 변경은 추적해야 했고 민감한 정보는 모든 도구에 똑같이 열어 두지 않았습니다.

— 페이지를 이렇게 늘리지 않을 수도 있었겠지. Bob이 말했습니다.

— 그랬을지도 몰라. 하지만 전에도 모순은 있었어. 단지 우리 머릿속에 있어서 아무도 보지 못했을 뿐이야.

Alice는 동료가 마지막으로 수정한 내용을 다시 읽었습니다.

— 원칙은 승인할게. 독일 고객에 관한 표현은 안 돼. 아직 아는 게 충분하지 않아.

Bob은 그 문장을 수정했습니다.

보증인의 일은 흔히 이런 모습이었습니다. 모든 화면을 승인하는 것이 아니라 가설이 무심코 규칙이 되는 일을 막는 것입니다.

다른 팀에서는 Thomas가 최근 합류한 개발자와 대화를 마친 참이었습니다.

Thomas는 ERP의 오래된 코드를 설명하는 데 한 시간을 썼습니다. 개발자는 옛 급여 메커니즘을 이해하기 위해 도우미를 사용하다가 한 예외가 왜 일반 규칙과 모순되는 듯한지 물었습니다.

Thomas는 과거 출시 기록에서 그 예외의 이유를 찾아냈습니다.

— 요약만 보면 코드를 정리해야 한다고 생각할 수 있어요. 그가 말했습니다. 하지만 정리하기 전에 무엇을 없애려는지 이해해야 합니다.

— 이 내용을 팀 컨텍스트에 넣어 주실 수 있나요?

— 지금 그러고 있어요.

Thomas는 새 메모를 공유했습니다. 출처와 설명, 테스트 사례 두 개를 추가했습니다.

그는 여전히 Lucas처럼 기술 변경 사항의 생성을 에이전트에게 맡기지 않았습니다. 회의적인 태도도 사라지지 않았습니다. 그래도 새로운 문서 공간은 기꺼이 사용했고 자신의 경험을 전하기 위해 다른 사람과 더 많이 일했습니다.

새 개발자의 적응을 지켜본 Vivien은 조금 뒤 Thomas에게 화상 전화를 걸었습니다.

— 어떻게 되어 가나요?

— 잘하고 있어요. 호기심이 많고 뭔가를 지우기 전에 질문합니다.

— 안심이 되네요.

— 왜 특정 검사가 존재하는지도 마침내 적을 시간을 냈어요.

Vivien은 미소를 지었습니다.

— AI에 관한 생각을 바꾼 건가요?

— 아니요. 이런 설명을 혼자 간직하는 것에 대한 생각을 바꿨습니다.

8

그는 잠시 멈췄습니다.

— 하지만 이런 일에는 시간이 필요해요. 다른 사람을 도우라고 하면서 제 하루를 개발 작업으로 계속 꽉 채우면 불가능합니다.

— Hubert와 이야기했어요. 계획에 멘토링 시간을 따로 확보했습니다. 다른 모든 일이 끝난 뒤에만 하는 활동으로 두지 않을 거예요.

Vivien은 자신이 무슨 말을 하는지 잘 알았습니다. 수년 동안 팀이 '시간이 조금 생기면' 신입을 교육하겠다고 말하는 모습을 보았습니다. 그런 순간은 오지 않았습니다.

두 사람은 개발자에게 필요한 것을 더 논의한 뒤 통화를 마쳤습니다.

Vivien은 출시 현황표를 한 번도 열지 않았습니다.

점심시간에 Hubert가 Vivien에게 통화를 요청했습니다.

— 질문이 하나 있어요. 그가 시작했습니다. 작년에 하던 일을 지금은 얼마나 하고 있나요?

Vivien은 생각했습니다.

— 예전의 제 직책을 풀타임으로 유지할 만큼은 아니에요. 그걸 묻는 거라면요.

Hubert는 바로 답하지 않았습니다.

Vivien은 오랫동안 두 기술팀이 반드시 거치는 사람이었습니다. 출시와 의존 관계, 긴급 상황의 일부, 개발자와 현업 책임자 사이의 논의를 관리했습니다. 조직의 자율성이 더 낮았을 때는 그 역할만으로 하루가 충분히 찼습니다.

이제는 그렇지 않았습니다.

주제마다 책임자가 명확했습니다. 개발자는 현장 결정의 상당 부분을 직접 내렸습니다. 정보는 Vivien이 전달하지 않아도 흘렀습니다. 도구는 진행 상황을 준비하고 의존 관계를 알려 주었지만 주장하는 내용은 여전히 확인해야 했습니다.

사람 사이의 문제와 갈등, 지원은 사라지지 않았습니다. Vivien은 계속 해결을 도왔습니다. 하지만 일상의 출시 감독을 전담하는 하나의 직책에 집중시킬 필요는 없어졌습니다.

— 쓸데없는 회의가 줄어든 건 괜찮아요. 오히려 좋아요. 그가 말을 이었습니다. 하지만 팀 관리를 배우는 데 이 모든 세월을 보내고 에이전트가 표를 제대로 채웠는지 확인하는 사람이 되고 싶지는 않아요.

— 그런 일을 부탁하지 않을 겁니다. Hubert가 대답했습니다.

두 사람은 전환을 준비했습니다. 중요한 기술 결정은 관련 보증인에게 돌아갔습니다. 지원을 맡은 사람에게는 별도의 시간이 주어졌습니다. Hubert는 단순한 문서에 맡길 수 없는 경력 선택과 인간적인 어려움을 계속 책임졌습니다. Vivien은 남은 조율 주제의 인수인계를 맡을 예정이었습니다.

완벽하지는 않았습니다. 어려움을 겪는 직원을 상대방이 돌보고 있다고 두 사람이 서로 믿었던 상황을 바로잡아야 했습니다. 책임자를 지정하지 않아 한 팀이 판단을 지나치게 오래 기다린 적도 있었습니다.

그때마다 책임과 그것을 보이게 하는 방식을 수정했습니다.

— 한 가지 원하지 않는 게 있어요. Hubert가 말했습니다. 당신의 직책을 없애고 그동안 한 일이 아무 가치도 없었다고 말하는 겁니다.

— 저도 원하지 않아요.

— 그렇다면 그 경험을 이제 어디에 활용할지 찾아야 합니다.

Vivien은 자신의 잔을 바라보았습니다.

— 아이디어가 하나 있는 것 같아요.

8

며칠 전 아내가 독일 이야기를 다시 꺼냈습니다.

그녀는 독일인이었습니다. 얼마 전부터 두 사람은 독일로 이주할 가능성을 논의했습니다. 서두르지는 않았습니다. 프랑스에는 삶과 습관, 친구들이 있었습니다. 그래도 이 가능성은 점점 더 구체적인 모습으로 계속 돌아왔습니다.

Vivien은 언제나 질문을 미뤘습니다.

Malorean에서 하는 일을 너무 좋아했습니다. 팀과 그 역사, 소프트웨어와 고객을 알았습니다. 떠나면 자신이 함께 만든 무언가를 버리는 기분일 것 같았습니다.

하지만 도전도 좋아했습니다. 경력의 이 시점에서 새로운 도전을 하고 싶어 하는 자신을 발견했습니다.

지난 토요일, 두 사람은 사실을 인정하지 않으려는 듯 독일의 주택 광고를 함께 보았습니다. 그러다 아내가 컴퓨터를 닫았습니다.

— 있잖아. 그녀가 말했습니다. 이 이야기를 앞으로 10년 동안 더 할 필요는 없어.

Vivien은 웃었습니다.

하지만 월요일에도 계속 생각했습니다.

— 독일이요? Hubert가 되물었습니다.

Vivien은 고개를 끄덕였습니다.

— 전략 회의에서 수년 동안 이야기했잖아요. 독일에 고객이 몇 곳 있지만 주로 프랑스 그룹의 현지 법인이에요. 독일 기업에 ERP를 파는 법은 아직 제대로 배우지 못했어요.

— 그렇죠.

— 모두가 제품이 준비되면 진출할 수 있다고 생각합니다. 하지만 현지 기업과 진지하게 이야기하기 전에는 무엇이 부족한지 절대 알 수 없어요.

Hubert는 주의 깊게 들었습니다.

— 시장 개발을 맡고 싶은 건가요?

— 네. 하루아침에 독일 시장 전문가가 된 척하겠다는 건 아닙니다. 우리 제품과 제약, 팀, 고객의 도입 방식을 알아요. 복잡한 문제를 중심으로 사람들을 모을 줄도 압니다. 나머지는 배워야 해요.

— 영업을 해 본 적 있나요?

— Julie나 Philippe처럼은 아니죠. 그들에게 일을 가르칠 생각도 없고요. 독일의 필요를 이해하고 우리가 대응할 수 있는지 알아보기 위해 그들과 배포팀, 현업 전문가들과 함께 일하고 싶습니다.

Vivien은 망설였습니다.

— 제 아내는 독일인이에요. 얼마 전부터 그곳에 가서 사는 이야기를 하고 있어요. Malorean을 떠나지 않고도 정말 하고 싶은 일을 할 기회가 될 수 있습니다.

Hubert는 잠시 말이 없었습니다.

— 단순한 번역 프로젝트가 아니라는 건 알죠?

— 물론입니다. ERP는 국가마다 다른 규칙과 프로세스를 다룹니다. 화면 번역만으로는 부족해요. 실제로 무엇을 약속할 수 있고 무엇을 개발하거나 조정하거나 거절해야 하는지 알아야 합니다.

— 틀릴 수도 있고요.

— 일부는 아마 틀릴 겁니다. 아무도 원하지 않는 기능에 1년을 투자한 뒤보다 고객과 함께 발견하는 편이 낫습니다.

Hubert는 프로젝트 목록을 바라보았습니다.

오랫동안 독일은 나중을 위한 포부였습니다. 먼저 밀린 주제를 끝내고, 개발자를 더 채용하고, ERP를 조정할 시간을 만들고, 시연을 준비하고, 팀을 교육해야 했습니다.

이 어려움의 일부는 사라지지 않았지만 여러 가지는 규모가 달라졌습니다.

시연 준비에는 노력이 덜 들었습니다. 매번 개발자를 동원하지 않고도 제품의 변형을 시험할 수 있었습니다. 지원팀과 사전 영업팀은 문서를 더 쉽게 만들고 지식을 공유했습니다. 제한적인 가능성을 탐색할 개발 역량도 전만큼 부족하지 않았습니다.

일이 없어졌다는 뜻은 아니었습니다.

프랑스 고객 요청의 대기열을 처리하는 것 이외의 일에 에너지 일부를 쓰기로 선택할 수 있다는 뜻이었습니다.

— 첫 단계를 제안해 주세요. Hubert가 말했습니다. 시장 진출 약속은 아니에요. 만날 사람, 답하려는 질문, 필요한 역량, 계속할지 판단할 방법을 담으세요.

Vivien은 미소를 지었습니다.

— 독일을 위한 Mini PRD인가요?

— 놀리지 마세요. 기술 주제를 그런 방식으로 정리할 수 있다면 시장 프로젝트의 가설도 적어 볼 수 있겠죠.

— 좋아요. 하지만 우리가 틀렸다는 걸 발견할 권리는 남겨 두고 싶습니다.

— 당연히 그래야죠.

다음 주 Hubert는 Alice와 Bob, Vivien, Julie, Philippe를 모았습니다.

그는 개발해야 할 기능의 수부터 묻지 않고 제안을 소개했습니다.

— 두 달 전에는 없던 역량이 이제 생겼습니다. 그중 일부를 독일 시장에 실제로 진출한다는 것이 무엇을 뜻하는지 이해하는 데 쓰고 싶어요.

Julie가 먼저 반응했습니다.

— 독일어 시연은 제가 이미 준비할 수 있어요. 잠재 고객에게도 꽤 빨리 연락할 수 있고요. AI가 있으면 몇 시간 만에 자료를 조정할 수 있어요.

— 자료는 그럴지 모르죠. Philippe가 말했습니다. 하지만 ERP를 사는 독일 경영자가 발표 자료가 자기 언어라는 이유만으로 선택하지는 않을 겁니다.

Julie는 팔짱을 꼈습니다.

— 그런 말은 안 했어요.

— 알아요. 공급업체를 어떻게 고르고 누구와 상의하며 어떤 신뢰를 필요로 하는지 이해해야 한다는 뜻입니다. 아직 그 시장을 충분히 몰라요.

대화가 도구에 관한 논쟁이 되기 전에 Vivien이 끼어들었습니다.

— 제가 배우고 싶은 것이 바로 그거예요. Julie, 가능성과 시연을 빨리 준비하는 역량이 필요합니다. Philippe, 큰 고객과 예쁜 화면만으로 해결할 수 없는 대화를 해 온 경험이 필요하고요. 배포팀은 고객이 너무 늦게 발견하곤 하는 질문을 알고 있습니다.

Alice가 말했습니다.

— 프로덕트 측에서는 한계를 아주 분명히 해야 해요. 빠르게 탐색할 수는 있지만 검증하지 않은 규제 기능을 약속할 수는 없습니다.

Bob은 동의했습니다.

— 어떤 기업을 먼저 이해할지도 골라야 해요. 독일 전체가 하나의 고객 유형은 아닙니다. 모두에게 답하려 하면 여기에서 피하는 법을 배운 실수를 반복하게 됩니다.

Hubert는 화면 속 얼굴들을 바라보았습니다.

Vivien에게 접근 방식을 제안해 달라고 했을 뿐입니다. 영구적인 팀을 즉시 만들거나, 모든 개발 역량을 예약하거나, 영업 시작일을 약속하라고 한 사람은 없었습니다.

그런데도 여러 사람이 이미 자신이 기여할 수 있는 것을 찾기 시작했습니다.

Malorean에서는 그것이 습관이 되어 있었습니다.

8

— 좋습니다. Hubert가 말했습니다. 이렇게 결정하죠. 첫 탐색 단계에 자금을 지원하겠습니다. Vivien이 맡습니다. 우리가 이미 해결할 수 있는 독일의 문제, 조정이 필요한 문제, 맡지 않을 문제를 알려 줘야 합니다. 그런 다음 더 투자할지 선택하겠습니다.

그는 덧붙였습니다.

— 그리고 도구 덕분에 시간을 아꼈다는 이유만으로 누군가를 이 프로젝트에 배정하지는 않겠습니다. 지금 지원하는 고객을 포기하지 않고 각자가 실제로 얼마나 시간을 낼 수 있는지 살펴봅시다.

Vivien은 마지막 내용을 적었습니다. 계획을 오랫동안 관리해 온 그는 문서상 확보된 역량이 현실에서도 언제나 가용한 것은 아니라는 사실을 알았습니다.

며칠 동안 그는 사전 영업팀과 배포팀 사람들을 차례로 만났습니다.

재사용 가능한 설정으로 시연을 준비하고 고객 대화를 더 쉽게 요약할 수 있게 되면서 그들의 업무 일부도 달라졌습니다. 두 사람은 같은 자료를 매번 다시 만드는 시간이 줄었고 독일에서의 첫 인터뷰에 참여하고 싶다고 말했습니다.

— 배포에 관한 질문을 준비하는 데 도울 수 있어요. 한 사람이 제안했습니다. 몇 차례 원격 미팅에도 함께하고요.

— 프랑스 고객은요? Vivien이 물었습니다.

— 모든 걸 놓을 수는 없어요. 하지만 계획에 넣으면 시간을 낼 수 있습니다.

반대로 까다로운 배포 여러 건에 투입된 다른 사람은 시간이 전혀 없다고 했습니다.

Vivien은 강요하지 않았습니다.

그들은 임시 분담안을 만들었습니다. 독일 기업의 기대를 이해하기 위한 시간 몇 개를 따로 확보하고, 도움이 될 때는 이미 고객인 현지 법인과 만나며, 실제 노력을 측정하기 전에는 장기적인 투입을 약속하지 않기로 했습니다.

절약한 시간이 저절로 성장으로 바뀌지는 않았습니다. 그 시간으로 무엇을 할지 결정하고, 되찾은 시간을 과거의 요청으로 곧바로 채우지 않는 선택이 필요했습니다.

저녁에 Alice는 Lucas에게 메시지를 받았습니다.

독일 프로젝트를 봤어요. 프로세스 차이 몇 가지를 시험하는 데 도움이 될 설정 메커니즘이 이미 있어요. 표준 제품을 건드리지 않고 탐색 환경을 준비할 수 있습니다. 관심 있으세요?

Alice는 메시지를 다시 읽었습니다.

두 달 전이었다면 비슷한 제안은 개발자가 주제를 맡겨 줄 때까지 기다려야 하는지를 두고 논쟁을 일으켰을 것입니다. 이제 첫 질문은 더 단순했습니다. 전략에 유용한가, 어떤 틀 안에서 진행할 수 있는가?

Alice는 독일 첫 단계의 컨텍스트를 열었습니다. Vivien은 가설과 책임자, 한계, 열린 질문으로 이미 채워 두었습니다.

Alice는 답했습니다.

그래. *Vivien*과 범위를 정하고, 업무 가설을 검증 없이 표준 동작으로 만들지 않는다는 조건이야. 우리가 만들 수 있는 것보다 배우려는 것에서 시작해.

Lucas는 거의 즉시 답했습니다.

알겠습니다. *Mini PRD*를 준비할게요.

Alice는 미소를 지었습니다.

Lucas가 이틀 만에 개발한 휴가 모듈을 처음 보여 준 금요일을 떠올렸습니다. 당시 모두가 어떻게 그렇게 많은 것을 그렇게 빨리 만들 수 있었는지 이해하려 했습니다.

이제는 Lucas가 시연을 빨리 만들 수 있다는 사실에 아무도 놀라지 않았습니다.

중요한 것은 왜 그것이 필요한가였습니다.

Vivien은 독일 기업들과 예정된 첫 만남의 날짜를 아내에게 알려 주며 하루를 마쳤습니다.

아직 계약을 하나도 맺지 않았습니다. Malorean은 사무실을 열지도 않았고 ERP를 시장의 모든 요구에 맞게 조정하지도 않았습니다. 현지 관행을 발견하고, 법률·규제 제약을 확인하고, 회사를 모르는 기업을 설득하는 법을 배워야 했습니다.

확실한 것보다 모르는 것이 훨씬 많았습니다.

— 무서워? 아내가 물었습니다.

— 조금.

— 팀을 조율할 때는 그렇게 자신 있어 보였는데.

Vivien은 미소를 지었습니다.

— 그 일은 수년 동안 배웠으니까.

— 지금은?

그는 첫 방문을 위해 작성하기 시작한 문서를 바라보았습니다.

질문. 가설. 만날 사람들의 이름. 자신이 모르는 것들의 첫 목록.

— 지금은 다시 배울 것이 생겼어. 그가 대답했습니다.

아내는 화면을 보려고 다가왔습니다.

— 그러면 우리는?

— 이사를 진지하게 이야기할 수 있게 됐지.

아내는 그의 어깨에 손을 올렸습니다.

Vivien은 과거의 출시 현황표와 월요일 회의, 프로젝트를 하나로 묶기 위해 했던 모든 일을 떠올렸습니다. 그는 그 일을 좋아했고 그 세월을 후회하지 않았습니다.

팀이 더 이상 같은 방식으로 자신을 필요로 하지 않는다는 사실을 받아들인 것도 후회하지 않았습니다.

화면에는 독일에서의 첫 미팅이 확정되어 있었습니다.

Malorean에게 아직 정복도 성공도 아니었습니다.

하나의 출발이었습니다.

12 – 맺음말

직군의 경계는 유연해지고 책임은 더 분명해진다

직군은 더 이상 각자가 할 수 있는 일을 정확히 구분하지 않습니다. 개발자는 필요를 탐색할 수 있고, 프로덕트 매니저는 무언가를 만들 수 있으며, 영업 담당자는 프로토타입을 만들 수 있습니다. AI는 과거에 다른 전문 분야에만 열려 있던 통로를 엽니다. 그렇다고 각자의 전문성이 서로 바꾸어 쓸 수 있는 것이 되지는 않습니다.

우리는 더 이상 자신이 직접 실행한 일만 책임지지 않습니다. 한 사람은 여러 기능을 가로지르는 주제를 맡을 수 있고, 보증인은 각 영역의 일관성을 지킵니다. 행동할 역량은 이동하지만 책임은 흩어져서는 안 됩니다.

위험은 두 가지입니다. 계속해서 '내 직무가 아니다'라고 말하거나, '할 수 있으니 혼자 결정해도 된다'고 생각하는 것입니다. 영역을 지키는 것만으로도, 영역을 없애는 것만으로도 업무를 조직할 수는 없습니다.

자율성은 공유 컨텍스트를 기반으로 한다

더 많은 자유는 각자가 판단에 필요한 것을 가지고 있을 때에만 의미가 있습니다. 공유 컨텍스트가 없으면 하나의 결정이 현장에서는 옳아 보여도 더 넓은 필요와 선택, 제약에 모순될 수 있습니다.

따라서 문서화는 보고서를 쌓는 일이 아닙니다. 해결할 문제와 목표, 가설, 결정의 이유, 한계, 에스컬레이션 규칙에 접근할 수 있게 하는 일입니다. 이런 요소를 형식화하면 단계마다 정보를 가진 사람을 다시 찾아 나서지 않고도 행동할 수 있습니다. 컨텍스트는 사람과 에이전트 모두를 위한 의사결정 도구가 됩니다.

여기서 보증인의 태도도 달라집니다. 모든 시도를 검사하는 것이 아니라 틀을 이해할 수 있게 만들고, 일관성을 지키며, 결정이 그 틀을 넘어설 때 개입합니다. 안전장치가 없으면 자유는 혼란이 될 수 있고, 모든 것을 승인받기 위해 위로 올려야 한다면 자유는 약속에 불과합니다.

관리도 함께 변합니다. 추적과 조율의 일부는 다른 방식으로 맡을 수 있습니다. 그래도 지원해야 할 사람과 전수해야 할 경험, 해결해야 할 갈등, 책임져야 할 판단은 남습니다. 과거에 이런 책임에 형태를 부여했던 작업이 사라져도 책임 자체는 사라지지 않습니다.

중요한 일을 더 많이 하기

이 변화를 생산한 기능의 수나 절약한 시간으로 측정하고 싶은 유혹이 클 것입니다. AI가 실제로 더 많은 것을 생산하게 할 수 있습니다. 하지만 아낀 시간을 곧바로 새 작업으로 채우면 목적을 묻지 않고 과거의 업무 방식을 가속할 뿐입니다.

그 역량을 필요를 더 잘 이해하고, 가설을 검증하고, 기존의 것을 개선하고, 가치를 만들지 않는 일을 포기하는 데 쓸 수도 있습니다. 어느 것도 저절로 일어나지 않습니다. 자유로워진 관심과 수단을 어디에 투자할지 선택해야 합니다.

Malorean Corp.의 Vivien은 예전 업무의 비중이 줄었다는 사실만 발견한 것이 아닙니다. 성공이 보장되지 않은 다른 문제를 위해 자신의 경험을 사용할 수 있다는 사실을 발견했습니다. 진보는 자동화한 것에만 있지 않고, 그 변화가 가능하게 한 것에도 있습니다.

이야기의 시작에서 모두는 Lucas가 그렇게 빠르게 만드는 것을 보고 놀랐습니다. 마지막에 질문은 달라졌습니다. 왜 만들고, 누구를 위해 만들며, 어떤 결과를 감수할 것인가?

AI의 핵심은 끝없이 더 많은 일을 하게 만드는 데 있지 않습니다. 정말 중요한 일을 더 많이 할 수단을 주는 데 있습니다.