Malorean Corp.

自分を重ねられる小説

理解するための全体像

開発者、AI、そして小さな企業：ほとんど本当の変革の物語

SaaS ERP のチームが
いま AI とともに
実際に経験していること



LAURENT DESÉCHALLIERS

Malorean Corp

> 少し前までは、一つのアイデアを試作品にするだけでも、何週間もかかることがあった。今では人工知能に助けられた一人の人間が、わずか数日で実現できる。この変化は驚くべきものだ。しかし、本当の大変革は別のところにあるのかもしれない。私たちが一緒に働く、その方法である。

Laurent Deséchalliers

翻訳について

本版は、フランス語の原文をもとに AI が作成した機械翻訳です。

人間の翻訳者による校閲は行われていません。内容、構成、事例は原文に忠実ですが、言い回しがぎこちなく感じられる箇所や、原語の痕跡が残る箇所があります。

本書をフランス語だけでなく複数の言語で届けるために、あえて受け入れた代償です。表現に疑問がある場合は、フランス語版を基準としてください。

はじめに

AI について私が最も驚かされるのは、コードを書けることではない。ついこの間まで数日、あるいは数週間かかっていたことを、驚くほど短い時間で実現できるようになった、その速さだ。

アイデアが実現可能かどうかの議論が終わる前に、試作品ができてしまう。開発者はチーム全員の手が空くのを待たずに、一つの方向性を試せる。プログラミングを知らない人でも、自分の構想を最初の形にできる場合がある。もちろん、試作品が信頼できる製品になるとは限らないし、複雑さが消えたわけでもない。それでも、まず試せるというだけで、何が可能だと思えるかが変わり始める。

ところが、私たちの企業は、最初の解決策をこれほど速く生み出せる世界を前提に作られてはいない。

企業は、専門技能、使える時間、担当者間の引き継ぎ、仕事を前に進めるために必要な意思決定を軸に組織されてきた。こうした仕組みには、それぞれ理由がある。しかし、一人ひとりにできることが変わると、新しい疑問が生まれる。これまでと同じ順番で仕事を進める必要があるのか。同じ承認を待つべきなのか。同じ仕事を、従来と同じ職種だけが担うべきなのか。

これは開発者だけの問題ではない。プロダクトマネージャー、デザイナー、営業担当者、チームの責任者など、多くの人が新しい働き方を見いだしている。これまでは同僚に頼んでいた仕事の一部を、自分で直接できるようになった人もいる。仕事そのものの性質が変わっていく人もいる。単なる道具の問題に見えたことが、協働や意思決定のあり方にまで及び始めている。

結末まで分かっている革命が始まった、と宣言したくなるかもしれない。しかし、私はまだそこまでは言えないと思う。

仕事や責任の分担を新しく試す企業がある一方で、慎重に進める企業もある。それには十分な理由がある場合も多い。有望に見える実践はあるが、日々の現実に耐えられるのがどれなのかを判断するには、まだ経験が足りない。小さなチームに向いたやり方が、規模の大きい企業や規制の厳しい企業、長い歴史を持つ企業にも、そのまま適しているとは限らない。

だから私は、AI時代に採用すべき正しい組織の作り方を示すマニュアルを書こうとは思わなかった。この本で試みるのは、本当に何が変わりつつあるのかを理解することだ。新しい道具で何が可能になり、企業がどう対応し、何を得て、何を失うおそれがあるのか。そして、これから何を選び取らなければならないのかを考えたい。

そのために、二つの道筋を交差させることにした。

一つ目は、観察だ。実在する企業、公表された実践、研究、さまざまな試みを見ていく。すでに確認できたことと、まだ仮説にすぎないことを区別し、魅力的な発表を実証された成果と取り違えないようにしたい。

二つ目は、物語だ。架空の企業 Malorean Corp.で働く人たちを追っていく。新しい可能性が日々の仕事に何をもたらすのか、彼らは少しずつ知っていく。組織を作り変えるための完成された計画があるわけではない。彼らには顧客がいて、仕事の習慣があり、差し迫った用事や信念がある。そして意見が食い違うこともある。フィクションは何かを証明するものではない。ただ、変化をそこで働く人の目線から見ることができる。

この二つの視点は、どちらも欠かせない。実例は、直感を事実と取り違えないために役立つ。物語は、組織の変化が組織図だけで決まるものではないと教えてくれる。信頼、技能、責任、そして一人ひとりの居場所にも関わるのだ。

結局、本書を通じて考え続けたいのは、次の問いである。**企業で働く人たちが、自分の職種ではこれまでできなかったことまで、はるかに多く実現できると気づいたとき、その企業には何が起きるのだろうか。**

著者について

私は数十年にわたってソフトウェア業界で働き、現在はプロダクトマネージャーを務めている。開発やシステム設計から出発し、チームの組織づくりを経て、プロダクト、その運用、そしてマーケティングへと活動の領域を広げてきた。

こうした経験のおかげで、人工知能の進化と、それが私たちの仕事に与える影響を間近で見ることができる。日々の実務のなかだけでなく、同僚のプロダクトマネージャーたちとの対話を通じても、その変化を実感している。私たちは体験や発見、疑問を持ち寄る。そうした会話から、道具の話を超えた問いがしばしば生まれる。私たちの職種で本当に変わっているのは何か。判断の仕方や、互いに協力する方法はどう変わるのか。

観察と対話を重ねるうちに、疑問は増えていった。いったん距離を置き、自分たちの印象をほかの企業で起きていることと照らし合わせ、その考察を分かち合いたいと思った。こうして本書が生まれた。

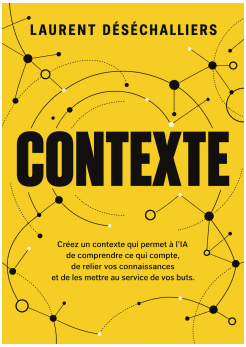
書くことは、以前から私の仕事に寄り添ってきた。専門誌に寄稿し、M2 Édition や Eyrolles では技術書の査読にも携わった。現在は malorean.net で、プロダクトマネジメント、AI、仕事の進め方、情報システムについて執筆している。

また、戦略・実務マーケティングを扱う独立系雑誌*Proposition Marketing*にも寄稿している。マーケティングや企業経営の分野から来た読者に向けて書くことは、共通の専門用語があると決めつけず、プロダクトの問題を分かりやすく伝える姿勢につながっている。

Laurent Deséchalliers

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

同じ著者の本



Le Contexte — AI が大切なことを理解し、あなたの知識を結びつけ、学んだことを目標のために積み重ねていける「コンテキスト」の築き方。

刊行：*2026* 年 *10* 月



NO AI — 理解を *AI* に委ねるな 記憶を保ち、思考を深め、批判的精神を鍛えるための、AI を使わない方法。自分自身の力を伸ばし、理解を AI に委ねることなく、AI をよりよく活かすための実践。

刊行予定：*2026* 年末

詳しくは https://malorean.net.

謝辞

まず、日頃から意見を交わしているプロダクトマネージャーの皆さんに感謝したい。私たちはプロダクトマネジメントやソフトウェア開発の仕事がどう変わっていくのか、さらに広く、AIが仕事と企業組織にどのような影響を与えるのかについて、互いの考えをぶつけ合っている。ときには熱を帯びるこうした対話は、絶えず新しい問いや考え直すきっかけ、発想をもたらしてくれる。その一部は本書にも反映されている。皆さんの好奇心、共有してくれた経験、議論を楽しむ姿勢に心から感謝する。

そして妻のChristineにも感謝したい。最初の本『Le Contexte』を書き終えたとき、私はしばらく休むと約束した。ところが、ほとんど間を置かずにこの二冊目に取りかかってしまった。幸い今回は、前の本ほど二人の夜や週末を費やさずに済んだ。Christine、いつも辛抱強く支えてくれてありがとう。私の考えごとが、また一冊の本になってしまうときでさえ。

事例についての注記

本書の考察と事例には、私がITのさまざまな職種で積んできた30年以上の経験が生かされています。そのもとになったのは、さまざまなプロジェクトや職場で経験した出来事に加え、他社のプロダクトマネージャー、CTO、エンジニアリングマネージャーなどとの数多くの対話です。つまり、複数の組織と、私の経歴のさまざまな時期にまたがっています。

本書で説明に用いるため、こうした状況には変更を加え、一般化し、組み合わせ直しています。ユースケース、数値、指標は、繰り返し生じる問題と、そこから私が得た学びを説明するためのものです。いずれも説明のために構成したものであり、実在する顧客、製品、組織を描いたものではありません。また、私の雇用主のデータ、慣行、見解を反映するものでもありません。

本書で述べる分析、方法、意見は私個人のものであり、私以外の誰かの立場を示すものではありません。

この本は AI で書かれたのか

はい。全編にわたり、人工知能と協力して書いた。

この問いが今、特別な意味を持っていることは分かっている。記事や小説、本が AI を使って書かれたのかが話題になる。人間が書いた部分と機械が書いた部分をどうしても区別しなければならないかのように、文章を AI 検出ツールにかける人もいる。

私は隠さずに話したい。資料調査から執筆まで、アイデアの整理、参考文献の確認、見落としている論点の洗い出しも含め、本書のすべての工程に AI が関わっている。しかし、これだけでは誰がこの本を考えたのかは分からない。だから、その協力の仕方を説明しておきたい。

かつて、これほどの規模の仕事を進めるには、資料を集める人、考えを検討する人、文章を読み直す人、構成や体裁を整える人などの力を借りることができた。今の私にとって、AI はこうした能力の一部を利用できるようにしてくれる。作業が速く、表現を提案し、矛盾を指摘し、ときには私が思いつかなかった質問を投げかける。自分に使える時間だけでは、一人で到底たどり着けなかった範囲まで考えを広げられる。

この本を書くため、私は ChatGPT の技術を基盤とし、Notion に情報を保持するコンテクストという仕組みを使った。資料、アイデア、決定、執筆途中の版を保存し、別の会話でも取り出せるようにしている。この方法は最初の本『Le Contexte』ですでに紹介した。そこでは、こうした仕組みによって AI が、いわば非常に有能な専属秘書のように働けることを説明している。AI は質問に答えるだけでなく、複数のプロジェクトの経緯を覚え、長期にわたって前に進める手助けをしてくれる。

ただし、ここには欠かせない区別がある。**仕事を実行できることと、理解して選択する責任を負うことは別だ。**この本を導く問い、そのもとになった私の職業経験、解釈や判断、そして最終的に何を支持し何を退け

るかは、私自身のものだ。AI は私に異議を唱え、考えを問い直し、より適切な言葉を探す手伝いができる。しかし、私の判断や著者としての責任に取って代わるものではない。

また、この本と、書くという行為そのものが創作の核心となる作品は区別している。François-Victor Hugo の翻訳で Shakespeare を読むとき、あるいは Cervantès や Cizia Zykë を原文で読むとき、作者の声、文章のリズム、文体は、語られる内容と同じほど重要だ。翻訳作品では、訳者の選択も読書体験の一部になる。そのような文学に AI をどこまで関わらせたいかについては、別の問いがある。ここで結論を出すつもりはない。

本書にはフィクションの部分もある。しかし主な目的は、技術と組織に起きている変化を理解することだ。読みやすく、心地よい文章であるべきだが、何より伝えたい考え、事実、問いに役立つ形でなければならない。

AI があったからこそ、仕事の合間に自分一人の力ではおそらく達成できなかった規模で、この企画に取り組めた。考えることの大切さを手放さずに、本書のような個人のプロジェクトを進められる。

そして考えてみれば、AI がもたらす変化を扱った本を、まさにこうした方法で書いたのは、どこか筋の通ったことでもある。

Laurent Deséchalliers

目次

翻訳について 3

はじめに 4

著者について 6

同じ著者の本 7

謝辞 8

事例についての注記 9

この本は AI で書かれたのか 10

Malorean Corp. 13

1 Malorean Corp. — いつもと違う一日 14
2 AI が開発者のソフトウェア生産能力を大きく高めた 19
3 Malorean Corp. — その翌日 26
4 新しい組織のあり方が現れ始めている 37
5 Malorean Corp. — できることと、すべきこと 45
6 役割の境界は曖昧になっても、責任は残る 57
7 Malorean Corp. — Alice が書き残してこなかった知識 65
8 自律性を高めるほど、明確なルールが必要になる 79
9 Malorean Corp. — 二つの妥当な判断 92
10 AI によって企業の中間管理職は消えるのか 104
11 Malorean Corp. — 取り戻した時間 116
12 おわりに 130

Malorean Corp.

1 – Malorean Corp. – いつもと違う一日

Malorean Corp.の本社は、フランスのル・アーヴルにあった。しかし新型コロナウイルスの流行以降、その住所だけでは社員が実際にどこで働いているのか、ほとんど分からなくなっていた。社員はフランス各地に散らばっていた。画面越しに顔を合わせ、共有画面を見ながら話し合う。仕事を進めるのに、同じ部屋にいる必要はほとんどなかった。

Malorean Corp.が開発していたのは、オンラインで使える ERP、企業のあらゆる業務に入り込んでいくような統合業務ソフトウェアだった。顧客には、社員が数人で、具体的な業務上の悩みを抱える小さな会社もある。一方、複雑な業務手順や独自の要件を持ち、曖昧な対応を容認しない大企業グループもあった。

顧客の大半はフランス国内だった。ドイツやスペインにも多少の利用者はいたが、主にフランスですでに顧客となっているグループの現地子会社だった。経営者の Hubert Félix Duca は、その先を見ていた。Malorean Corp.を欧州で本格的に成長させたいと考えていたのだ。

難しいのは、ERP が企業紹介サイトのように簡単に国境を越えられる製品ではないことだった。国によって法律や規則が違う。給与計算の慣行も、行政上の義務も、企業の仕事の進め方も異なる。Hubert はそうした制約をよく知っていた。彼の出身分野は経営管理、会計、財務であって、ソフトウェア開発ではない。彼にとって機能に価値があるのは、顧客が困らずに使える場合だけだった。

社内の役割分担は比較的明確だった。

開発チームが二つあり、それぞれに五人の開発者がいた。二人のプロダクトマネージャー、Alice と Bob が、顧客のニーズを理解し、役立つ改善を選び、製品の方向性を示すために両チームと働いていた。Alice は元開発者で、技術的な議論に入っても全体像を見失わなかった。Bob はビジネススクールの出身で、違う角度から課題に取り組んだ。経歴は違っていたが、二人はとてもうまく協力していた。

営業チームは新規顧客を開拓し、既存顧客との関係を維持していた。マーケティング担当者が一人、それを支えていた。さらに五人の専門家が、販売前の段階から導入時まで、顧客先または遠隔で支援していた。ただ ERP をインストールするのではない。顧客企業が必要としていることを明確にし、業務手順を適応させ、販売後の六か月間も支援を続けることが多かった。

サポートチームには十二人いた。七人が小規模顧客、五人が大口顧客を担当していた。ある機能が顧客の役に立つどころか、ソフトウェアを迂回する手作業を強いていると最初に気づくのは、多くの場合彼らだった。二人の法務担当者は Malorean Corp.の法務を担う一方、給与計算のような慎重さが必要な規則に関わる機能を設計するときには、製品づくりにも専門知識を提供していた。

これで会社は動いていた。もちろん、完璧ではない。多くの顧客に導入されたソフトウェアには、次々と要望が寄せられる。例外も増え、変えにくい習慣も残る。作り直す時間がないために使い続けている機能もあった。それでも、誰もが自分のやるべきことをおおむね理解していた。

そこへ、ChatGPT という言葉が社内の会話に登場し始めた。

当初、大規模な変革計画があったわけではない。好奇心の強い二人が数か月前から使い始めていただけだ。営業の Julie Morel と、開発の Lucas Martin だった。二人は自分の仕事で試し、何ができるようになったかを周囲に話すようになった。

Julie にとって、その利点はすぐに分かるものだった。難しいメールを準備し、もっと分かりやすい表現を求め、商談前にメモを整理し、いくつかのアイデアをプレゼンの構成にまとめる。スライドの内容まで作ることもあった。顧客に何を伝えるかを決めるのは ChatGPT ではない。材料を渡し、提案を見直し、表現を修正し、自分で責任を持てる内容だけを送っていた。

この道具が自分以上に顧客を理解しているなどとは、Julie も思っていなかった。ただ、すでに自分が知っていることを文章にまとめるだけで、三十分も費やさずに済むのがうれしかった。

同僚の Philippe Garnier は、少し距離を置いてそれを眺めていた。大口顧客を長年担当してきた経験豊かな営業担当者だった。契約書の向こうにいる人たちを知っていた。報告書には残らない意見の食い違いも、会議の意味を変えてしまう何げない一言も知っていた。

「メールだけでも使ってみたら？」と、ある朝 Julie が勧めた。「文章を整えるだけでも便利よ」

「メールなら自分で書けるよ」

意地悪で言ったのではない。ただ、自分の仕事の一部をなぜ道具に任せるのか、理由が分からなかった。自分の価値は顧客を知っていることと、その顧客に合わせた話し方にある。そう考えていたからだ。

Julie はそれ以上勧めなかった。次の商談の準備があった。

Lucas は、ChatGPT をまったく別の使い方で試していた。リヨンに住む経験豊かな開発者で、単純作業を数分短縮したいだけではなかった。何時間も実験し、ますます難しい質問を投げ、答えが気に入らなければやり直した。Julie が何より実用的な結果を求めていたのに対し、Lucas はどこまで可能なのかを探っていた。

やがて初期利用者たちの感想が Hubert の耳にも入った。一人当たり月二十ユーロほど。面倒な作業で少しでも時間を取り戻せるなら、十分に元が取れるように思えた。そこで Malorean Corp. は全社員に ChatGPT を使えるようにした。

盛大なプレゼンで宣言するような革命ではなかった。役立つと思えば誰でも使える。毎日開く人もいれば、めったに使わない人もいる。Philippe は一度も使わなかった。ほとんどの人にとって、AI はあくまで助手だった。何をしたいか伝える。AI が提案する。人が確かめ、最後に決める。

会社の日常は続いていた。

水曜日、プロダクトと開発の両チームが話しているとき、以前からの課題がまた持ち上がった。休暇管理の機能だ。

誰もこの機能をあまり気に入っていなかった。古く、融通が利かず、選べることも少ない。使い続けられる程度には動いていたが、苦情がなくなるほどにはうまく動いていなかった。

休暇を申請したい社員は、職場のパソコンを使わなければならなかった。日曜の夜、休みの予定を思いついても、携帯電話から気軽に申請することはできない。もっと困るのは休暇日数の計算だった。祝日を正しく考慮できなかったのだ。月曜日から金曜日まで休みを申請し、水曜日が祝日の場合、本来は四日と数えるところを五日と数えてしまうことがあった。

サポートもその問題を知っていた。営業も知っていた。顧客のなかには、休暇管理機能に繰り返し不満を抱く会社もあった。使われていたのは、そこに機能が存在するからであって、期待に応えているからではなかった。

「きちんと作り直す必要があるね」と誰かが言った。

異論はなかった。しかし「きちんと作り直す」とは、画面の設計をやり直し、日数計算のルールを修正し、スマートフォンでも使えるようにして、長年のあいだに古くなった部分を置き換えるということだ。

二か月ほどの開発になると考えられた。ほかにも優先事項はある。この案件は、またしても「いつかやるべきこと」の一覧に戻された。

金曜日の午後、オンライン会議が終わりかけたところで、Lucas が数分だけ時間をほしいと言った。

「ちょっと見てもらいたいものがあるんです」

何を見せるかは予告していなかった。水曜日の話し合いのあとで Lucas が休暇管理機能を作っていたとは、誰も知らなかった。

会議画面に Lucas の共有画面が映った。すっきりとした新しい画面が開く。見やすく、現代的で、必要な情報があるべき場所に置かれていた。

Lucas は休暇申請の手順を見せた。それからウィンドウを携帯電話ほどの幅まで狭めた。画面が自動で切り替わる。そのまま操作して日付を選び、申請を確定できる。デスクトップのパソコンに戻る必要はなかった。

次に、月曜日から金曜日までの一週間を選び、水曜日を祝日にした。

表示された休暇日数は、四日だった。

日付を変えると、計算結果も変わった。携帯電話用の表示に戻り、いくつかの画面をスクロールし、次々と操作してみせた。静止した画面デザ

インを並べただけではない。入力欄は反応し、計算が走り、一連の手順が実際に機能していた。

初めのうちは、同僚たちも普通の質問をしていた。あの項目はどこにあるのか。日付を変えたらどうなるのか。やがて質問が止まった。

ビデオ会議の小さな画面に映る表情が変わった。

水曜日には二か月かかると話していた。金曜日には Lucas が動く最初の版を見せている。誰にも助けを頼んでいない。誰にも知らせていない。そして今のところ、どうやったのかも説明していない。

「待って」と、ようやく誰かが言った。「それ、一昨日話していた休暇管理の機能？」

Lucas はうなずいた。

沈黙が流れた。

誰もが画面に映る同じものを見つめていた。何週間も計画して進めるはずだった仕事が、たった二日で形になっている。

どうしてそんなことが可能だったのか、誰にも分からなかった。

2 – AI が開発者のソフトウェア生産能力を大きく高めた

何十年もの間、ソフトウェア開発の組織は、一つの単純な制約を中心に形づくられてきた。**ソフトウェアを作るには大きなコストがかかる**、という制約だ。

アイデアを製品に変えるには、時間、希少な専門技能、多くの調整が必要だった。一つの機能についても、目的を定め、作業を分け、優先順位を決め、開発し、テストし、レビューし、展開しなければならない。どれだけの仕事に取り組めるかは、使える開発者の人数、経験の深さ、そこに割ける時間によって、自然と限られていた。

この希少性が組織の形を決めてきた。開発者の時間を守り、作業の準備を前もって済ませ、割り込みを減らし、バックログを管理し、実装を始めるずっと前から何を選ぶか議論するようになった。現代のプロダクトマネジメントの多くも、こうした条件の下で発展した。高価な開発リソースを、見当違いのアイデアに費やさないためだった。

コードを書く AI アシスタントが登場し、さらに広い作業を実行できるエージェントが現れたことで、この前提が変わり始めている。

もはやコード補完が上手になるだけではない

開発者向けの初期の AI ツールは、生産性を高める道具の新しい世代として理解できた。関数の続きを書き、コードを説明し、テストを生成し、リファクタリングを提案する、といった使い方だ。

次の世代では、仕事の性質が大きく変わる。エージェントは一つの仕事を受け取り、コードベースを調べ、複数のファイルを変更し、テストを実行し、一部の不具合を修正して、人間がレビューできる変更を提出する。開発者は作業の大部分を委ねられ、何より、以前より多くの仕事を同時に動かせるようになる。

結果は単に「開発者が速くコードを書ける」ということではない。

一人が着手し、前に進められるソフトウェア開発の仕事量が、飛躍的に増える。

すでに目を引く成果を発表する企業もある。AI を中心としたエンジニアリングの仕組みへ移行したと説明する AMPECO は、開発速度が四倍になり、ストーリー当たりのバグが 50％減ったと報告している。ただし、これは市場全体の平均ではない。特定の状況にある一社が公表した数字だ。それでも、目指している変化の大きさを示している。関数を書く時間を数分短くするだけでなく、チーム全体の実行能力を大きく伸ばそうとしているのだ。[1]

企業の発表だけが手がかりではない。私自身も Claude Code でそれを体験した。四日間、180 回を超えるコミットを重ね、画面、API、テストを備えた製品の最初の版を作った。以前の方法なら一〜二か月はかかっただろうと見積もっている。2026 年 7 月に Malorean.net でその体験を書いた。もちろん、そこから生産性の向上率を一般化することはできない。過去に開発経験のある PM が何を作れるようになったか、そして生成コードの品質、保守性、管理にどんな難しさが残るかを示す例だ。[7]

この点で、問いの規模が変わる。

AI によるコード作成の効率向上が数％程度なら、組織はそのままでも効果を吸収できるだろう。しかし生産能力がはるかに速く伸びると、ほかの制約の重要性が相対的に増す。

コードを速く作れても、企業全体が速くなるとは限らない

ソフトウェアチームは、コードを書く工程だけでできた一本の生産ラインではない。

作り始める前には問題を理解し、投資に値するか判断し、方向性を選び、制約を明らかにする必要がある。作った後にはレビュー、テスト、安全性の確認、展開、利用状況の観察、保守がある。その間にも判断は続く。

このうち一工程だけが飛躍的に速くなっても、ほかの工程に流れ込む仕事が増えるだけだ。

顧客理解が同じ速さで進まなくても、コードは速く作れる。どの試作品を育てるべきかを決める能力が増えなくても、試作品の数は増やせる。

レビューに届く変更が増えても、それを確認する人間の注意力まで同じ比率で増えるわけではない。

だから、AI によってほかの仕事が急に不要になるのではない。

仕事のボトルネックが別の場所に移るのだ。

以前は実装の遅さに隠れていた問題が、表に出てくる。選んだ問題が適切か、目的が明確か、どう判断するか、何を検証するか、全体の整合性を保てるか、リリース後に学べるか、といったことだ。

たくさん作ることは、たくさん価値を生むことではない

この区別は、ますます重要になる。

作ること自体が難しかった時代には、リリースできたソフトウェアの量が、組織の価値創出能力に近いものと見なされがちだった。もともと不完全な見方だったが、AI の時代には危険にさえなる。

チームは、コードも、プルリクエストも、試作品も、機能も、以前より大量に生成できる。それでも製品が良くなるとは限らない。むしろ、欠陥、やり直し、コスト、保守すべき複雑さが増えれば、組織全体の成果を損なうことさえある。

言い換えれば、

作れることと、作る価値があるものを見極められることは違う。

そのため活動量の指標だけでは、ますます判断できなくなる。変更の数、コードの行数、AI ツールを使ったという事実だけでは、チームが価値を増やしたかどうかは分からない。

Canva は、注目すべき例を示している。Product、Engineering、Data の各チームが実験の事業的な影響を見積もり、現場の判断とビジネス上の結果をより直接的に結びつける仕組みを作った。[2]

すると、問いそのものが変わる。

以前は、

私たちはどれだけ作れるのか。

これからは、

作れるものがこれほど増えたなかで、本当に作る価値があるのは何か。

> **ソフトウェアの生産性という問題には、四十年以上の歴史がある**
>
> *1986* 年、*Fred Brooks* は**No Silver Bullet**を発表し、*1987* 年に**IEEE Computer**誌に掲載した。彼はソフトウェア製作に伴う**偶発的な難しさ**（プログラミング言語、道具、設計をコードに移すこと）と、システムを理解し、設計し、組み合わせ、進化させる際の**本質的な複雑さ**を区別した。単独の技術革新や管理手法だけで、十年間に開発の生産性、信頼性、単純さを十倍にできるとは期待できないと論じた。*[3]*
>
> この論考は、*1975* 年に初版が出版された**The Mythical Man-Month**の *1995* 年記念版にも収録された。*[4]* 生成 *AI* は実装できる量を大きく変える可能性がある。それでも、選ぶ問題が適切になるとは限らない。これは現在から *Brooks* の考えを捉え直したものであり、**彼が *AI* エージェントの登場を予言したという意味ではない**。

新たなボトルネックは人間の判断力

実装のコストが下がるほど、顧客についての知識、問題の理解、戦略上の背景、目的を明確に表現する能力、リスク評価、製品を見極める感覚、断る力、決定に対する責任の価値は、相対的に増す。

2026 年 3 月、私は Malorean.net の記事で同じ考えを述べた。技術的な道具が手に入りやすくなるほど、競争上の優位は、顧客、業務、利用者、技術的制約、競争、規制など、その状況の理解へと移る。これは戦略的な分析であって、実験で実証した結果ではない。「判断力を働かせる」とは、単に作り方を知るのでなく、何を作るのが正しいか選べるほど状況を理解するということだ。[8]

AI は「どう作るか」を身近にする。

しかし「なぜ作るか」が自然に明らかになるわけではない。

組織が問題を十分理解する前に、AI は解決策を作れてしまう。会社の市場に合わないと判断すべき顧客の要望も、すばやく実現できてしまう。本当に良い成果につながるか分からないまま、一つの機能を何通りにも作り分けることができる。

作るのが簡単になるほど、「作らないもの」を選ぶ規律が大切になる。

Ramp の事例はこの原則をよく示す。同社の CPO は、優先すべき理想的な顧客像（ICP）と、特に優れた形で解決したい課題を明確にし、製品を別の方向へ引っ張る要望には、たとえ収益性が高くても断る必要があると強調している。[5]

2026 年 7 月、Marty Cagan はこうしたずれを「AI 生産性のパラドックス」と名づけた。より速く作れても、そもそも作る価値のなかったアイデアの問題は解消しない、という見立てだ。彼は、学ぶために作るものと、成果を生むためにリリースするものを区別する。これはプロダクト分野の実務家による分析であって、すべての企業が同じ影響を受けることを証明する測定結果ではない。[6]

AI のパラドックスは、つまりこういうことだ。

行動する能力を大きく高めるからこそ、何をするか選ぶことが以前より重要になる。

道具ではなく、仕組み全体の変革

この変化への対応として、開発者全員にコード支援ツールを配り、あとはこれまでどおり働くよう求めることもできる。

それでも、個々の現場では効果がある。

しかし、その効果が大きくなるほど、組織は別の限界にぶつかる。理解し、判断し、調整し、確認し、生み出されたものを受け止める能力だ。

この段階で、問題はもはや道具を導入するかどうかだけではない。

組織の問題になる。

エンジニアリングの能力が最も希少だった時代に設計された組織が、その希少性が薄れた後も最適であり続ける理由はない。

そこで避けて通れない問いが生まれる。

希少な資源である開発能力を守るために設計された組織は、その能力がはるかに豊富になったとき、どう変わるのだろうか。

参考文献

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

- URL： https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- AI エージェントを中心にエンジニアリング組織を組み直した経験と、同社が主張する能力向上を紹介する。

[2] Measuring Commercial Impact at Scale – Canva

- URL： https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- プロダクトと開発の実験を、事業への推定影響と結びつけて把握する Canva の仕組みを説明する。

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

- URL：https://www.cgl.ucsf.edu/Outreach/pc204/NoSilverBullet
- 1986 年発表、1987 年公刊の論考。ソフトウェア開発の本質的な複雑さと偶発的な難しさを区別する。

[4] The Mythical Man-Month, 1995 年記念版、第 16 章 – Fred Brooks

- URL： https://www.oreilly.com/library/view/mythical-man-month-the/0201835959/ch16.xhtml
- *No Silver Bullet*が後の版に組み込まれたことを確認できる出版社の案内。章の一部のみを公開している。

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

- URL：https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- Ramp の CPO による、集中すべき顧客と問題、製品の方向性を散らす要望を断る重要性についての説明。

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

- URL：https://www.svpg.com/the-ai-productivity-paradox/
- AI による開発速度の向上と、実際に生み出す価値の違いを論じた 2026 年 7 月の実務家による分析。因果関係の実証ではない。

[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.en.html
- Claude Code による個人的な製品開発の体験。速度、技術的な問題、テスト、品質管理を扱う。統制実験でも一般化可能な証拠でもない。

[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.en.html
- 顧客、業務、規制、技術の背景を理解することが競争優位の中心になるという著者の戦略的考察。定量的な実証ではない。

3 – Malorean Corp. – その翌日

「AI を使ったんです」

Lucas は、まるでその一言ですべて説明できるかのように、ごく自然に言った。

まだ画面を共有していた。新しい休暇管理機能がウィンドウの大部分を占めている。数分前には、携帯電話で休暇を申請する方法と、祝日を考慮した日数の計算を実演していた。

「ちょっと待って」と Alice が聞き返した。「AI を使ったって、コードを書くのを手伝ってもらったということ？」

「そうです。でも、それだけじゃありません。機能の設計、既存コードの理解、画面の開発、計算、テストの作成……ほとんど全部ですね」

「それを水曜日からやっていたの？」

「水曜の午後からです。最後の作業は今朝終えました」

Alice は画面に並ぶ同僚の顔を見た。笑みを浮かべる人もいた。何を見落としていたのか考えているような人もいた。

「でも仕様書はなかったでしょう」

「はい。水曜日に話したことは覚えていました。あの機能と問題点は知っていましたから。まずAI に既存の仕組みを分析させて、それから一つずつ進めたんです」

「全部動くの？」

Lucas は小さく笑った。

「全部かどうかは分かりません。今見せたところは動きます。まだ確認することはあると思います」

同じチームで働く Thomas が、初めて口を開いた。

「裏のコードはどうなってる？」

「あとで見ていいよ。テストもかなり書いた」

「見てみるよ」と Thomas は答えた。

特に感心した様子はなかった。あるいは、感心していても顔には出したくなかったのかもしれない。

Hubert Félix Duca は、それまで黙っていた。

十分快調に動くデモを何度も見てきた彼は、それが何百もの企業で使える製品であるとは限らないと知っていた。例外的なケースも、大口顧客の固有の要件も、導入して初めて表に出る問題も知っていた。

それでも、水曜日に聞いた数字を覚えていた。

二か月。

今目の前にあるものは、彼らが想像していたものにすでによく似ていた。

「Lucas、それに何時間くらいかけたか、もう一度教えてくれる？」

「だいたい二日です」

Hubert は画面に少し身を乗り出した。

「そうか」

それ以上の言葉は出てこなかった。

会議は数分後に終わった。みな自分の仕事に戻ったが、その日の午後に予定していた業務の話は、誰もあまりしなかった。

土曜日の朝、Alice はコーヒーをいれながら前日のデモを思い返していた。

気になったのは Lucas の速さだけではない。誰の助けも必要としなかったことだった。

Alice はプロダクトマネージャーとしての仕事をよく理解していた。開発を始める前には課題を調べ、要望を比較し、顧客と話し、開発者と一緒に妥当な解決策を探す。

休暇管理のような案件なら、何度かの話し合いを設定しただろう。計算ルールを明確にし、例外をどう扱うか判断し、何から開発すべきか決める手助けをしたはずだ。

Lucas はそのどれもしていなかった。

話し合いを聞き、自分で動き、解決策を作った。

見事だと思う気持ちがあった。同時に、すべての開発者がこうして働き始めたらどうなるだろう、という不安もあった。

とはいえ、この数か月で彼女自身の日常も大きく変わっていた。

最初、Alice は AI を議事録の要約や文書の書き直しに使っていた。その後、より大きな仕事を任せるようになった。顧客からのフィードバックの処理、いくつかの報告書の準備、似た要望の整理、以前は何時間もかかっていた事務作業を一部自動化した。

以前なら午前中のかなりの時間を取られていた仕事が、今では数分で終わることもある。

そうした作業がなくなっても、少しも惜しくなかった。

むしろ、その逆だった。

顧客と話し、要望の奥にある問題を理解し、本当に何を作るべきか考える時間が、ようやく増えたのだ。

AI は彼女の仕事をつまらなくしたのではない。

仕事をうまく進める妨げになっていたものを、いくつか取り除いてくれた。

それなら、開発者だけが今までとまったく同じ働き方を続けなければならない理由は何だろう。

Alice はカップを置いた。

問いはとても明快だった。

答えはそうではなかった。

Hubert も週末のかなりの時間を考えごとに費やしていた。

会社にはいつも、実行できる量を上回るアイデアがあった。何か月も待っている機能があり、顧客に求められても後回しにされ続ける改善があった。欧州展開の構想もあり、ソフトウェアをまだ大きく改修しなければならなかった。

障害は分かりきっていた。全部に取り組む時間が足りない。

金曜日、Lucas はその障害が以前ほど絶対的ではないかもしれないと示した。

二か月かかると見積もった仕事が、二日で最初の版になるなら、ほかにどれだけの企画が可能になるのだろう。

Malorean Corp.には十人の開発者がいた。Hubert は十人全員が同じ速さで仕事を進める姿を想像した。

次に、十人がそれぞれ調整もせず、好きな機能を作り始める姿を想像した。

二つ目の光景は、あまり愉快ではなかった。

土曜日のうちに Alice へ電話しようかとも考えたが、思いとどまった。

日曜の夜、短いメッセージを送った。

明日の朝、少し時間はある？　金曜日のことをもう一度話したい。

数分後に返事が来た。

あります。私も話したいです。

月曜日の朝九時、Alice と Hubert はビデオ会議で顔を合わせた。

「それで？」と Hubert が聞いた。

「それで、とは？」

「金曜日に何が起きたか、分かったかい？」

Alice は笑った。

「技術的には、だいたい。Lucas は AI を使って、私たちが想像したよりずっと速く開発しました」

「それ以外は？」

「まさに、それが問題です」

Hubert はうなずいた。

「週末じゅう考えていたよ。何年も開発能力が足りない。顧客の要望を断り、企画を先送りにし、やらないことを選んでばかりいる。ところが Lucas は、二か月と見積もっていたものを二日で作ってみせた」

「最初の版です」と Alice は念を押した。「本番環境に出せるかは、まだ分かりません」

「分かっている。でも、仕上げにあと二週間必要だとしても、大きな違いだ」

Alice にも反論はできなかった。

「それに、気になることがあるんだ」と Hubert は続けた。「彼は君を通さずに全部やった」

Alice はすぐには答えなかった。

「私も、それが気になりました」

「仕事の進め方をもう一度教えたほうがいいと思うかい？」

「そう考えました」

「それで？」

「でも、それは間違いだと思うんです」

Hubert は眉をひそめた。

「とはいえ、開発者が何を作るか一人で決め始めたら、困ったことになるかもしれない」

「もちろんです。ただ、今回のことを考えてください。水曜日に休暇管理を作り直すべきだと話しました。いつもどおりなら、私はまず要望を集め、会議を開き、案件を整理して仕事を細かく分けたでしょう。何かを見せるまでに何週間もかかったかもしれません」

「それが全部無駄だとは限らない」

「そうです。顧客のニーズを理解するのをやめようとは言っていません。ただ Lucas は、そのすべてを待たずに進められると示したんです」

「つまり、開発者が自分で決めればいいと？」

「いえ、そうではありません」

Alice はしばらく考えた。

「私たちは二つのことを混同している気がします。何をすべきか決めることと、どう実現するか決めることです」

「二つはつながっている」

「はい。でも、だからといって、すべての判断に私が入る必要はありません」

Alice は少し姿勢を正した。

「この数か月で、私の仕事も大きく変わったんです。AI を毎日使っています。手作業でしていたことをたくさん自動化して、書類の準備や情報整理に使う時間はずいぶん減りました」

「そう聞いているよ」

「でも、どれほど変わったかは、まだ伝わっていないかもしれません。今は本当に大切なこと、顧客を理解し、問題を見つけ、私たちの力をどこに使うか決めることに、ずっと多くの時間を割けます」

Hubert は耳を傾けていた。

「以前の私の仕事には、案件を社内で前に進めることも含まれていました。チケットを書き、依頼を具体化し、質問に答え、人と人を調整する。もし明日、開発者がこうしたことを私にほとんど頼まなくなったら？」

「君の時間が増える」

「そうです。開発者たちの時間も」

Alice はひと呼吸置いた。

「AI が私の仕事を変えるのは受け入れるのに、なぜ開発者の仕事だけは変えたくないのでしょう？」

Hubert は椅子の背にもたれた。

「分かった。でも、何を作るかは誰かが決めないといけない」

「当然です。何のために作るのかは、もっと大事です」

「じゃあ、どうしたい？」

Alice はためらった。

プレゼンの準備をしていたわけではない。提案できる新しい組織モデルもなければ、導入したいプロセスもない。

あるのは、直感だけだった。

「一つ、試してみてはどうかと思うんです」

Hubert は続きを待った。

「Lucas に一つ案件を任せるんです。実際の顧客の問題を。そして解決方法には、今までよりずっと自由を与えます」

「仕様書なしで？」

「完成した答えを渡さないんです。問題を説明して、その問題を知っている人と話せるようにして、自分で調べてもらいます」

「君はどうする？」

「いつでも相談に乗ります。ニーズを理解する手伝いをし、彼のアイデアに疑問を投げかけ、作れるからという理由だけで何かを作ってしまわないよう支えます。でも、すべての手順を私が指示するわけではありません」

Hubert は考え込んだ。

「つまり、プロダクトマネージャーの仕事の一部を彼にやらせる？」

「そうかもしれません。あるいは、二人の仕事をいつでもきっちり分けるのをやめるだけかもしれません」

「うまくいかなかったら？」

「そのときは分かります」

Hubert は笑った。

「それは少し心もとない方法だね」

「分かっています」

Alice も笑った。

「でも、新しい組織の話し合いに三か月かけるより、小さな案件で試すほうがいいと思います」

Hubert はすぐには答えなかった。

それから何かを思い出したように言った。

「ちょうど、食品業界の顧客から何週間も指摘されている問題がある」

Alice はうなずいた。すぐに思い当たった。

Malorean Corp.の ERP は在庫と製造工程を管理していたが、完成品に使われた原材料のロットを正確に追跡できなかった。

食品メーカーでは、仕入先から不良原材料のロットについて連絡があったとき、それを使った製品をすぐ特定できなければならない。

現状では、顧客が自分たちで情報を調べ直すしかなかった。

「先週もその話があったんだ」と Hubert は説明した。「ある顧客は、仕入先から警告が来て、対象製品を突き止めるために品質管理責任者と丸一日かかったそうだ」

「サポートからも似たケースが何件か来ています」

「Lucas にできると思う？」

Alice は考えた。

休暇管理よりずっと繊細な問題だった。画面を数枚作れば済む話ではない。工場で何が起きているか、原材料がどう使われ、顧客が製品をどう識別しているかを理解する必要があった。

だが、それこそが面白いと思った。

「問題を彼に任せてみていいと思います」

「業務を十分に知っているかな？」

「おそらく、まだ知りません」

Hubert は眉を上げた。

「それは安心だね」

「私だって細部まで知っているわけではありません。でも、誰に聞けばいいかは知っています。自社の制約をよく理解している顧客がいます。サポートや導入支援の人たちは、毎日その問題を見ています。Lucas が直接聞けばいいんです」

「顧客と直接話すというのか？」

「いけませんか？」

「普段は君の仕事だろう」

「だからこそです」

Hubert は、まだ完全には納得していないようだった。

「ニーズに合わないものを作ってしまったら？」

「そうならないよう手助けするのが私の役目です。ただし、五十ページの仕様書を書くのではありません」

「その解決策を本番環境に出したいと言ったら？」

「ほかの機能改善と同じように、確認と承認が必要です。自由に働けるようにするからといって、あらゆるリスクを取る権限まで与えるわけではありません」

Hubert はしばらく沈黙した。

「何がいちばん心配か、分かるかい？」とようやく口を開いた。「誰かが失敗することだけじゃない。誰が何をしているのか、分からなくなることなんだ」

「私も心配です」

「それでも試したい？」

「はい」

Alice は考えてから言い添えた。

「もう一つの道は、今までとまったく同じ働き方を続けて、開発者にもっと速くコードを書いてもらうだけです。でも金曜日の Lucas が見せたのは、それだけではなかったと思います」

二人はさらに二十分ほど話し合った。

Hubert は進行中の企画を混乱させたくなかった。Alice も、始める前から社内に大々的に発表するような実験にはしたくなかった。

そこで、やり方を簡単にした。

Lucas に原材料ロットの追跡という課題を任せる。解決策を探し、ニーズを理解するために必要な人と直接会う自由を与える。

Alice は彼のプロダクト面での相談相手になる。詳細なバックログを用意するのではなく、重要な決定には関わり続ける。

最初は範囲を絞り、顧客に納期は約束しない。

そして何より、その後どうなるかをきちんと観察する。

「どうなれば成功と呼べるのかは、決めたほうがいいね」と Hubert が言った。

「はい。開発した機能の数だけでは測りたくありません」

「何を見る？」

「まず、問題を理解して役に立つものを提案できるか。その次に、こちらがすべて段取りしなくても、どこまで進められるかです」

「こちら側の時間がどれだけ必要かもだね」

「それもです」

Hubert は時計を見た。

「よし、試そう」

Alice は笑ったが、完全に安心したわけではなかった。

「やってみましょう」

「今日、Lucas に話す？」

「はい」

会議は終わった。

Alice はしばらく画面の前に座ったまま、メッセージを開いて Lucas を探した。

書き始めた。

Lucas、今日少し時間ある？　いつもとは少し違う案件を提案したいんだけど。

送信する前に読み返した。

自分自身が具体的な解決策を思い描く前に、開発者へ問題を託すのは、久しぶりのことだった。

それが正しい考えかどうか、まだ分からない。

そして、彼より先に答えを知っていることが、自分の役割なのかどうかも、もう確信できなかった。

4 – 新しい組織のあり方が現れ始めている

AIによって、これまでより多くの仕事を実行できるようになった。では、企業はチームや意思決定の仕組みをどう組み直しているのだろうか。その答えは一つではない。

開発者を支えるプロダクトマネジメントを強化する企業もある。開発者の役割を広げる企業、技術職ではない人たちにも実行手段を与える企業、エージェントを使う小さなチームを中心に仕事を組み直す企業もある。

従来の分業が、唯一の当然の形ではなくなっている。

従来のモデルにも理由はあった。Product Managementが顧客と事業についての情報を集め、Designが体験を形にし、Engineeringがそれをソフトウェアにする。チケット、仕様書、引き継ぎ、会議を使って、限られた開発能力を各職種のあいだで調整していた。

この組織の形が、突然悪いものになるわけではない。

ただ、実行できる量が増えると、人、背景情報、仕事の分担を違う形で変え始める企業が出てくる。

今はまだ「正しいモデル」を探す段階ではない。

まずは、現れつつある主な対応を見ていこう。

開発者を支えるプロダクト部門を強化する

最も直感的なのは、ProductとEngineeringの比較的伝統的な区別は残したまま、開発者の周囲に配置するプロダクトマネジメントの人員を増やす方法だ。

Gammaが興味深い例を示している。同社の説明では、エンジニア約四人に対してプロダクトマネージャーが一人いる。よく引き合いに出される従来の比率は、エンジニア約十人に対して一人だ。[1]

このモデルでは、開発能力を伸ばすことと、プロダクトマネジメントを手厚くすることが同時に進む。

発想は分かりやすい。より多く作れるなら、それだけ多くの力を使って問題を理解し、優先順位を決め、仕事の方向性を示そうということだ。

開発者が案件のより広い範囲を担う

別の戦略は、エンジニアの担当範囲を広げることだ。

開発者は、問題が解決策やチケットに完全に翻訳されるまで待つ必要がなくなる。ニーズの理解、試作、現場での判断、結果の確認にも、より深く関わることができる。

Linear は Product Engineer の求人情報で、問題を定義する段階にも参加し、プロダクトマネジメントに常に依存することなく、一つの機能をアイデアからリリースまで担う役割を説明している。[2]

このモデルは、同じ人がより多くの情報と行動の手段を持つことで、職種間の引き継ぎを減らそうとするものだ。

開発者全員が PM になるという意味ではない。

一部の案件で Engineering の役割が広がるということだ。

職種の境界を越えて働けるようにする

この動きは開発者だけのものではない。

Figma の役割の変化に関する調査では、回答者の 64% が自分は少なくとも二つの役割に当てはまると答え、三分の一を超える人が複数職種にまたがる責任を担っていると報告した。この結果を AI だけの影響とすることはできないが、職種間の境界が以前ほど固くないことをよく示している。[3]

Atlassian は、デザイナーが実際の製品コードを直接編集し、Engineering によるレビューに先立って自分でプルリクエストを作る仕事の進め方も紹介している。[4]

Cursor も、Product、Design、Engineering が以前より実装に近いところで働く組織について説明している。[5]

したがって、起きているのは「全員がすべてをする」ということではなく、**従来の担当範囲を越えて仕事をする人が増えている**ということだ。

そこから別の問いが生じる。複数の職種が同じ領域で行動できるなら、責任をどう明確に保つのか。

この点には後で戻ろう。

少人数のチームに、より広い責任を持たせる

少ない人数でより広い仕事を担うという対応もある。

AMPECO が説明する「alignment syncs」では、エンジニアと Product Owner が目的をそろえ、その後の分析、開発、テストの多くをエージェントが担当する。[6]

Taito は、チームが計画を承認したあと、一人または二人がエージェントを使って実装を進める方式を紹介している。[7]

こうしたモデルでは、小さな人間のチームが、以前より多くの仕事を動かせる。

組織上の変化として見えるのは、職種間の引き継ぎが減り、一人または少人数のグループを中心に案件がまとまる可能性だ。

ただし、その人たちがどこまで独自に決められるのか、いつ分野の責任者に相談しなければならないかを定める必要は残る。

確認作業の一部を自動化する

生産が速くなると、検証と承認の仕組みにも圧力がかかる。

チェックをさらに自動化したり、変更のリスクに応じて扱いを変えたりする企業がある。

Meta は、コード変更が障害につながる危険性を推定する Diff Risk Score を使っている。[8]

Cloudflare は、マージ前に自動で確認できる Engineering の標準を明文化している。[9]

Taito も、決定的なルール、レビューを行うエージェント、リスクに応じたトリガーを組み合わせている。[7]

この段階で重要なのは、**すべての変更を同じ方法で検証する必要は、もはや必ずしもない**ということだ。

自動化、人間の責任、上位判断を求める基準をどう組み合わせるかは、後で詳しく取り上げる。

一つずつ指示するより、判断に必要な情報を渡す

さらに進んで、詳細なチケットを仕事の調整の中心に置くやり方を問い直す企業もある。

Linear は、PM が問題の範囲を決め、システムに書き込み、Engineering が実装するのを待つモデルを明確に批判している。顧客のフィードバック、アイデア、戦略上の方向性、決定、コードを、人間にもエージェントにもアクセスできるようにする必要性を訴えている。[10]

Cursor も、現場でより多くの判断ができるよう、十分に明確な目標を与えることを重視している。[5]

つまり、調整そのものがなくなるわけではない。

詳細な指示を渡す形だった調整の一部が、**より豊富で、より直接利用できる背景情報を共有する形**に移るのだ。

どんな情報を共有し、どう更新し、自律性にどんな制約を設けるべきかは、別の章で検討する。

ニーズをよく知る人が、自分で作れるようにする

実際のニーズを知る部署が、開発チームの順番待ちをせず、小さな道具を自ら作れるようにする企業もある。

Shopify は Quick という社内プラットフォームを作り、より多くの社員が Web サイトや小規模なツールを素早く公開できるようにした。[11]

Vercel は、技術職ではない社員が安全な環境で社内ツールを作り、開発者を主力製品の優先課題から引き離さずに済むモデルを紹介している。[12]

Atlassian のデザイナーが技術レビュー前にコードを編集する例も同じ流れにある。[4]

ここでは、AI が開発者を速くするだけではない。

実装を、**そのニーズの背景をすでに理解している人**へ移せる場合があるのだ。

AI 以前から存在する原則

Marty Cagan は 2018 年に、すでに*empowered product teams*という考え方を提唱していた。経営側がチームに渡すのは納品すべき機能の一覧ではなく、解決すべき問題だ。プロダクト、デザイン、エンジニアリングの技能を使って解決策を探し、生み出した量ではなく結果に責任を持つ、部門横断のチームである。[13]

彼は Chris Jones とともに、2020 年の著書*EMPOWERED*でこの考えをさらに発展させた。職種が交換可能になるとか、マネージャーが不要になるという話ではない。共通のビジョン、確かな専門技能、チームを支える責任者が必要だという前提に立っている。[14]

だから、この発想自体は AI エージェントより前からあった。AI によって変わり得るのは、自律的なチームが探ろうと決めた解決策を、より速く、より直接的に実行する力だ。**プロダクトチームの自律性という目標は以前からあった。AI は、その手段と制約を変えつつある。**

まだ一つに収束していない選択肢

こうした対応は、一つの変革の連続した段階ではない。プロダクト部門を強化する方法と、開発者により多くのプロダクト面の責任を任せる方法は、矛盾して見えるかもしれない。それでも、別々の企業ではどちらも成り立ち得るし、同じ会社の中で共存することさえある。

共通点は別のところにある。**実行しやすくなるほど、背景情報、意思決定、責任をどう分けるかが、改めて組織の選択になる**ということだ。

発表や組織図を離れて、この変化が働く人々の日常に何をもたらすかを見ていく必要がある。

次に Malorean Corp.へ戻り、こうしたモデルがフランスの小さな企業の日常をどう変えるのか見てみよう。その後、自律性を支える責任とルールを検討する。

参考文献

[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
- 開発能力が高まるなかで Gamma がプロダクトマネジメントを手厚くしている事例。

[2] Product Engineer – Linear Careers

- URL： https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- 問題の定義から提供までを担う Product Engineer の求人と職務内容。

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

- URL：https://www.figma.com/blog/2025-shifting-roles-report/
- 従来の職種の境界を越えた役割や責任の広がりを示す調査。

[4] Designers' workflow for shipping code – Atlassian

- URL： https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code
- デザイナーが製品コードを直接変更し、Engineering の確認を受けるまでの流れ。

[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
- Product、Design、Engineering が以前より密接に製品作りへ関わる様子。

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

- URL： https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- 少人数が目的を定め、その後の実装の多くをAIエージェントへ委ねる方式。

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

- URL： https://www.swarmia.com/blog/inside-an-ai-native-engineering-organization/
- 一〜二人がエージェントによる開発を進める、小さなチームの実践。

[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/
- 変更による障害リスクを推定し、検証手順を変える仕組み。

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

- URL： https://blog.cloudflare.com/engineering-standards-enforcement/
- 明確に書かれた技術標準を開発工程の自動チェックにつなげる事例。

[10] Issue tracking is dead – Linear

- URL：https://linear.app/next
- チケットだけでなく、顧客の声、戦略、決定、コードをつなぐ情報共有を重視する主張。

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

- URL：https://shopify.engineering/quick
- 社員が社内ツールを直接作って公開するためのプラットフォーム。

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

- URL： https://vercel.com/blog/stopping-the-slow-death-of-internal-tools
- 専門の開発チームの優先課題を妨げず、業務担当者が社内ツールを作るモデル。

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

- URL：https://www.svpg.com/empowered-product-teams/
- 機能の納品を指示されるチームと、課題の解決を任されるチームの違いを述べる。

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

- URL： https://www.svpg.com/books/empowered-ordinary-people-extraordinary-products/
- 自律的なチーム、目標、戦略的背景、リーダーによる育成を扱う著書の紹介。

5 – Malorean Corp. – できることと、すべきこと

火曜日の朝、Lucas は Alice を驚かせる質問から話を始めた。

「顧客と直接話してもいいですか？」

前日、サポートチームがロット追跡の問題について保存していた報告書を何件か受け取っていた。そこには製造番号、原材料、Excel ファイル、仕入先からの警告が書かれていた。Lucas はその夜、AI に手伝わせて情報を整理していた。

「もちろん」と Alice は答えた。「どうして？」

「何が足りないのか、まだ正確に分からないんです」

Alice は笑った。一週間前なら、自分が先に顧客と話して結論を開発チームへ渡しただろう。しかし今は、Lucas に案件を任せると約束している。

「その顧客を担当しているサポートの人を紹介するわ。面談を設定してもらいましょう」

こうして水曜日、Lucas は「ロットを管理する」という一言では収まらない複雑な状況を知った。

顧客企業の品質管理責任者が、仕入先から届いた文書を見せた。ある原材料を使用から外さなければならないという。仕入先は問題のロット番号を把握していた。製造会社も、どの発注分を受け取ったかは知っていた。しかし、その原材料を使った完成品を正確に特定するには、複数の書類を照合しなければならなかった。

「前回は四人がかりで、半日以上ファイルを確認しました」と責任者は説明した。「それでも確信が持てず、最後は現場まで戻ったんです」

Lucas は製造時に情報をどう記録しているか見せてほしいと言った。顧客は ERP の画面を開き、それから表計算ファイルを見せた。

「でも、なぜ同じ番号を Excel にも書いているんですか？」

「そちらのソフトウェアだけでは、知りたいことが分からないからです」

責めるような口調ではなかった。そのぶん、余計に耳が痛かった。

通話を終えたとき、Lucasの手元には三ページのメモがあり、最初より確信が減っていた。

「どうだった？」と、折り返しの電話でAliceが聞いた。

「最初は、ロット番号を登録する画面を作ろうと思っていました」

「今は？」

「実際に使った原材料をどう記録しているのか、まず理解する必要がありそうです。そこが分からなければ、きれいな画面ができても、顧客の質問には答えられません」

Aliceはうなずいた。月曜日に同じことを教えることもできた。しかし、Lucasが自分で気づいたのは悪くなかった。

翌金曜日、Lucasはデモ用のデータを使った最初の版を見せた。

原材料のロットを一つの製造記録に結びつけ、そこから生まれた完成品を追跡できた。警告の対象となったロットから、関係する製品を探す小さな例も用意してあった。

Aliceは何度かクリックした。

「今週、これを作ったの？」

「そうです。まだ間違いや足りないところは多いけれど、顧客に見せられるものができました」

「逆方向にも追えるようにしたら？」

「それもあります」

Lucasは別の画面を開いた。

Aliceは声を出して笑った。

「そうでしょうね」

休暇管理機能のデモ以来、彼女が企画の規模を考える感覚は変わっていた。以前なら次の四半期へ回したものが、今ではすぐ手の届くところにあるように思えた。

「仕入先で検索する機能も付けられるわね。対象の発注に直接警告を出したり、回収用の書類を作れたりしたら？」

Lucas はすぐには答えなかった。

「アイデアは書き留めましょう」と、ようやく言った。「でも、その前に今あるものを持って、もう一度顧客に会いたいんです」

「もう少し完成させるために、十五日も待つわけにはいかないでしょう？」

「デモそのものが心配なんじゃありません」

「何が心配なの？」

今度 Lucas が共有した画面は、あまり見栄えがよくなかった。古いテーブル、意味の分からない名前のフィールド、製造と在庫を結ぶ複数の関連づけが並んでいた。

「画面ならいくらでも作れます。問題は、うちの ERP がロットの履歴を全部残す設計にはなっていないことです。何年も前の在庫移動があって、手作業で直されたものもあります。そもそも記録していない情報もあります」

「それも AI を使って整理できないの？」

「移行プログラムを書くのは手伝ってくれます。でも、記録したことがない情報は、AI にも作り出せません」

Alice は黙った。

「ちゃんとやるなら」と Lucas は続けた。「データモデルの一部を変えて、ほかの機能への影響も確認しないといけません。二週間早くするために顧客の在庫を狂わせるわけにはいかないんです」

「ソフトウェアを壊せと言っているわけじゃないわ」

「分かっています。でも金曜日から、何か見せるたびに、Alice は新しい要望を三つ返してくるでしょう」

責めるような言い方ではなかった。それでも Alice には痛い指摘だった。

「そのとおりね」と認めた。

「僕だって速く進めたくなります。知っているでしょう。でも一時間でコードを生成できるようになったからといって、十五年間動いてきた ERP が簡単になるわけじゃありません」

少し迷ってから付け加えた。

「Thomas が休暇管理機能のコードの一部を見たんです。僕たちが間違って扱っていたケースを二つ見つけてくれました。彼が正しかった。直しましたけど、一つ思い出しました。作るのがずっと速くなっても、落とし穴を知っている人は必要なんです」

Alice はデモの日、Thomas がただ「コードを見せて」と言っていたのを思い出した。

「それで、どうしたいの？」

「架空のデータでデモを続けます。本当にニーズを理解できているか確かめたい。でも本物の在庫データを変更するなら、既存の仕組みを理解し、必要な検証をする時間がほしいです」

Alice は別の画面にまだ開いてある最初の版を見た。

先週、彼女は Hubert に対して Lucas の自律性を擁護した。ところが、その自律性には自分に「いいえ」と言う権利も含まれるのだと知って、少し複雑な気持ちになった。

「分かったわ」と、ようやく言った。「今は実データに触らないでおきましょう」

8

翌月曜日、Hubert は週次会議で、すっかりおなじみになった質問をした。

「それで、あの実験は？」

「進んでいます」と Alice。「予想より速いところも、遅いところもあります」

「どういうこと？」

Alice は Lucas のデモを見せ、金曜日のやり取りを説明した。

Hubert は技術的な懸念に関心を示した。

「慎重で正しいよ。ロット追跡の問題を解くために、全顧客の在庫に問題を作りたくはない」

「そうですね。私も少し急がせてしまいました」

「君が？」

「この二週間、何でもできるような気がしていたんです。アイデアが浮かぶと、すぐ試せると思ってしまって」

Hubert は笑った。

「最初のデモを見たあとの私と同じだね」

「問題は、画面に何かが現れるまでの速さと、信頼できるものができるまでの速さを混同しやすいことです」

「少なくとも Lucas は、いつ減速すべきか知っている」

「はい。その判断を信頼する必要があります」

二日後、Julie Morel が Alice に五分ほど時間はあるかと聞いた。

Alice はこの言い方に慣れていた。Julie との五分は、あっという間に三十分になり得る。

「先週、かなり大きな見込み客に会ったの」と Julie は切り出した。「工場をいくつも持っていて、それぞれの仕事の仕方に合わせられる ERP を探しているの。準備したものを見て」

画面に整ったモックアップがいくつも現れた。注文の種類に応じて異なる承認手順、拠点ごとに設定されたルール、利用者によって変わる画面が描かれていた。

「これ、あなたが作ったの？」と Alice。

「ChatGPT と一緒にね。商談のメモから操作の流れを提案してもらって、二人で直したの。一晩かかったけど」

Alice には、これらの画面が何に役立つかすぐに分かった。商談報告書の中に埋もれていた要望が、数時間で目に見える形になっている。

「顧客にも見せた？」

「まだ。先に Alice に見せたかったの。でも、こういうことができるなら商談が大きく変わると言っていたわ」

Alice は画面を順に見た。

「面白いアイデアね。Lucas に直接話してみたら？」

Julie は驚いた顔をした。

「直接？　Alice が案件をまとめてくれると思ってた」

「どうして？　顧客を知っていて、直接話して、画面まで作ったのはあなたでしょう。私よりずっと詳しく、相手が何を望んでいるか説明できるわ」

「Lucas なら、できるかどうか見てくれる？」

「ええ。でも Julie、気をつけて。実現可能か確かめることと、開発すると約束することは別よ」

「もちろん」

Alice は、この「もちろん」をよく知っていた。嘘ではない。ただ、契約の可能性を前にした営業担当者の高揚感だった。

翌日、三人で会議を開いた。Julie はモックアップを見せ、見込み客の要望を説明し、質問に答えた。

Lucas がすぐに注目したのは、拠点ごとにルールがどう違うかだった。

「似た仕組みの一部は、すでに ERP にあります」と言った。「まったく同じ目的ではないけど、あるんです。試せるものを作れそうですね」

「どれくらいかかる？」と Julie。

「デモだけなら、そんなに長くはないと思います」

Julie は Alice を見た。その笑顔は、もう契約を取った人のようだった。

「来週これを見せられたら、本当に有利になるわ」

Alice も、考える前に「進めよう」と答えたくなった。だが、ロット追跡の仕事、積み上がった要望、この見込み客だけの特別なルール、ほかの顧客には何の役にも立たないかもしれない機能が頭をよぎった。

「いいえ」と彼女は言った。

二人が振り向いた。

「何に対して、いいえなの？」と Julie。

「その契約を取るために、これを製品の機能として開発し始めることには賛成できない」

「でも面白いって言ったじゃない」

「そうよ。ただ、この顧客が必要としているのが、ほかの顧客にも役立つ機能なのか、それともその工場だけの特別なやり方なのか、まだ分からない」

「受注を逃すかもしれない」

「そうかもしれないわ」

Julie はこの返事を予想していなかった。

「今なら、こういう依頼にすぐ応えられるのよ」と食い下がった。「その力を使わないのはもったいないわ」

「使いたくないわけじゃない。何を提案しているのか理解したいの。特別な要望を全部 ERP の中心に組み込んだら、いつか手を入れられない製品になってしまう」

「でも Lucas はできるって」

「Lucas はデモを作れると言ったの。製品として提供すると言ったわけじゃない」

沈黙が流れた。Julie はもう一度モックアップを見た。

「じゃあ、何と伝えればいい？」

「課題を理解していて、可能な方法を一つ見せたい、と。製品へ機能を追加すると決めた、とは言わないで」

Julie はうなずいた。落胆しているようだったが、意欲を失ったわけではなかった。

「分かった。工場ごとの違いをもっと詳しく話してもらう面談を設定するわ」

「いいわ。モックアップはそのまま使いましょう。正しい質問をする助けになるから」

Julie が会議を退出したあと、Alice は Lucas と少しだけ残った。

「本当にすぐ作れた？」

「最初の版なら、はい」

「作ってみたいと思った？」

「正直に言うと、はい。楽しそうでしたから」

Alice は笑った。

「私も、いいと言いたかった」

「何が踏みとどまらせたんですか？」

「私たち二人とも、できることばかり話して、すべきことをほとんど考えなくなっていたと気づいたから」

翌週、Lucas から Alice へリンクとメッセージが届いた。

昨日ちょっと作ってみたんです。見てもらえますか？

Alice が開くと、いつもの ERP の画面があった。だが横に小さなパネルが追加され、いくつかのルールを変更できる。拠点を選び、承認が必要になる基準を変え、手順の表示・非表示を切り替え、デモをもう一度実行できた。

Alice は Lucas に電話した。

「これ、何？」

「ルールを変えてデモを準備するための小さなツールです。昔の設定機構を再利用しました。テスト環境でしか動きません」

「どれくらいかかったの？」

「二時間くらいかな。Julie のモックアップを思い出して、考えついたんです」

「でも、あなたが今担当している案件じゃないでしょう！」

「ええ。違う設定を試すたびにコードを書き換えずに済むか、見たかっただけです」

Alice は設定を変え始めた。画面が反応した。別の設定、それから三つ目も試した。

「待って。つまり、私一人で何種類ものデモを準備できるの？」

「公開してあるルールの範囲なら、はい。何でもできるわけではありません。それに、絶対に顧客の環境では使わないでください」

「でも Julie に二つの操作手順を見せて、どちらがあの見込み客に合っているか聞けるわね」

「そのとおりです」

Alice はしばらく画面を眺めた。最初の休暇管理機能は Lucas の仕事を速くした。このツールは、昨日まで自分にはなかった力を与えてくれた。

開発者にわざわざ頼まなくても、アイデアを試せるようになったのだ。

「これで何ができるか、想像できる？」

Lucas は笑った。

「たぶん。でもこれを完全な製品にしろと言われる前に、二時間の試作品だと念を押しておきます」

「約束するわ」

電話を切ったあとも、Alice は設定を変えて遊び続けた。

翌月曜日、Hubert は Alice がいつもの進捗管理文書を開いていないことに気づいた。

「今日は何も準備していないの？」

「準備しました。ただ、見せたいものがあります」

Alice は Lucas のツールを見せ、Hubert の目の前で同じデモの二つの版を作ってみせた。

Hubert はすぐ理解した。

「この変更を開発者に頼んでいないのかい？」

「はい。Lucas が小さなツールを作ってくれたんです。これでいくつかのルールを自分で試せます」

「営業チームにも使わせられないかな？」

「将来的には。でもまだ十分に頑丈ではありません。今は Julie が、見込み客のケースについて私と一緒に使えます」

Hubert は興味深そうに画面を見た。

「ロットの案件は？」

「Lucas が進めています。最初の版を持って顧客に会い、何が必要かもっと分かってきました。ただ、履歴データの扱いを安全にするまでは手を出さないと言っています」

「妥当な判断に聞こえる」

「ええ。金曜日は少しいらだちました。でも今は、彼が譲らなくてよかったと思っています」

Alice は指先でペンを回した。

「何か不思議なことが起きています、Hubert。Lucas が顧客と話して、解決策について判断している。Julie はほとんど使えるモックアップを持ってくる。そして私は、開発してもらわずに ERP のいくつかの案を試せるようになりました」

「不安かい？」

「ときどき。誰もがほかの職種の仕事を一部できるようになったら、私たちの職種はどうなるのだろうと思って」

「答えは見つかった？」

「まだ、はっきりとは」

Alice は少し考えた。

「ただ、すべてのアイデアが私を通らなければならない状態にはしたくありません。Julie がモックアップを私に渡して、チケットに変えてもらうのを待つ必要はなかった。Lucas に画面の作り方を一つずつ説明する必要もありません」

「でも見込み客の要望を開発するのは断ったね」

「はい。あれは私たちの製品についての決定だからです。私の判断が間違っているかもしれないし、考えを変えるかもしれない。でも、その選択に責任を持つのは私です」

「Lucas は在庫管理で近道するのを断った」

「何を壊しかねないか知っているからです。あの判断は、私が急いでいても尊重しなければならない」

Hubert は画面の小さな Alice の顔を見つめた。

「つまり、できることを増やしても、誰が判断するかを曖昧にしてはいけない、ということだね」

「そう思い始めています」

Alice はためらってから付け加えた。

「以前は、たくさんの案件が自分を通るから、自分の仕事が大切なのだと思うことがありました。でも今は、最後に何が実現するかのほうが大事なのかもしれません。正しい問題を解いているか。信頼できるものを作っているか。それを作る価値があるか」

Hubert はうなずいた。

「その問いに答えるべき人が誰なのか分からなければ、困るね」

「はい」

二人は数秒黙った。この会話を新しい組織の発表に変えたいとは、どちらも思っていなかった。

実験は始まったばかりだ。可能性が増えた一方で、決めるべきことも増えていた。

Alice に通知が届いた。Julie からだった。見込み客はデモのいくつかの案を見ることに同意した。あわせて、いつ自分たちの社員が使えるようになるのか知りたがっていた。

Alice はすぐには返事をしなかった。

ソフトウェアで答えを作ることは、今やずっと速くなっていた。

それでも、**どの答えを Malorean Corp. として約束し、その責任を負うのか**は、誰かが決めなければならなかった。

6 – 役割の境界は曖昧になっても、責任は残る

長い間、Product、Design、Engineering の境界は、仕事を分ける分かりやすい目印だった。

Product が問題を理解し、何を作るか決める。Design が体験を設計し、Engineering がその意図をソフトウェアに変える。現実には、これほどきれいに分かれたことはなかった。それでも、この区分がチーム、採用、プロセス、責任の形を決めてきた。

この分担は AI 以前から問い直されていた。1.4 章で取り上げた自律的なプロダクトチームでは、Product、Design、Engineering が共通の問題に取り組んでいた。今、AI は別の境界を越える動きを加速している。各職種の人が、以前はほかの専門職しか実行できなかった仕事を、より多くこなせるようになったのだ。

組織の事例を見ても、その変化はうかがえる。Figma の調査も一つの例だ。[1] 私も Malorean の記事で、開発者が問題の理解へ近づき、PM がその実装へ近づく、双方向の動きについて書いた。[2]

そこから職種が消えてしまうと考えたくなるかもしれない。しかし、私にはむしろ、一人ひとりの責任を定義し直す必要があるように思える。**職種の境界が曖昧になるほど、責任の所在は明確にしなければならない。**

行動できるのは、もう一つの専門職だけではない

AI が最初に広げるのは、必ずしも決定権ではない。行動する力だ。

Atlassian が紹介する、デザイナーが自分でコードを変更する事例を考えてみよう。技術チームは引き続き、品質と本番導入の条件を確認している。[3] これは昔の分業に戻ることではない。コードを変更する能力と、その信頼性への責任が別物であることを示している。

Linear の Product Engineer の求人にも同じ区別が見える。問題の発見から提供した結果まで、案件を一貫して担うことが求められる。[4] それで製品戦略のあらゆる決定権まで与えられるわけではない。

職種間の引き継ぎを減らしながら、その引き継ぎに意味を与えていた専門性は残せる。重要なのは関わる職種の数ではなく、**異なる職種の人が同じ領域に関わるとき、誰が何に責任を負うか**だ。

貢献できることと、その分野の責任を負うことは違う

企業が混同しがちな二つを分けなければならない。行動する許可と、責任である。

営業担当者が社内ツールを作れても、プラットフォーム全体のセキュリティ責任者になるわけではない。

PM が画面を変更できても、Design System の一貫性を守る責任を負うことにはならない。

開発者が機能の細部についてプロダクト上の判断を下せても、会社全体の戦略、ポジショニング、ロードマップまで責任を持つとは限らない。

この区別は多くのチームにすでに存在している。しかし AI は、それを明文化する必要性を強める。

高度に専門分化した働き方では、職種と責任が重なることが多かった。ある仕事を実行できる人が、その分野を担当する人でもあった。行動できる人が増えると、この重なりは崩れる。

そこで、自律性と責任の曖昧化を混同する危険が生まれる。

私は以前から、品質について似た原則を主張してきた。開発者が何かをリリースしたあと、その欠陥は QA チームやバグ対応当番、次に手の空いた開発者のものだと片づけるわけにはいかない。リリースしたものの技術的な品質には、責任を負う人がいるべきだ。[5]

職種の境界が曖昧になっても、この原則は変わらない。むしろ重要になる。

「Garant（保証責任者）」という考え方

この問題に対応するため、私は **Garant** という考え方を使っている。

重要な分野には、それぞれ整合性と品質に責任を持つ個人またはチームを置く。

Garant は、その分野の仕事をすべて実行する人とは限らない。

あらゆる細かな決定を承認する人でもない。

その分野に影響する重要な選択があるとき、制約を越えるとき、問題が起きたときに、組織が説明と対応を求める相手だ。

単純な組織なら、少なくとも三つの分野で責任を明確にできる。

- **プロダクトの Garant**：製品としての一貫性、重要な選択、対象顧客、価値提案、大きな判断、戦略との整合性。
- **デザインの Garant**：Design System、体験の一貫性、アクセシビリティ、新しい設計パターン、UX の品質基準。
- **開発の Garant**：アーキテクチャ、セキュリティ、性能、保守性、技術的品質、本番導入の条件。

原則は簡単だ。

誰もが複数分野に貢献できる。しかし、各分野には最後に責任を負う人が残る。

こうすることで、職種をまたいで働けるようにしながら、責任の所在を失わずに済む。

営業担当者が作っても、技術的な Garant になるわけではない

分かりやすい例を挙げよう。

営業担当者がニーズを発見し、AI を使う道具で小さなアプリや便利な変更を作ったとする。企業に十分安全な環境があるなら、Engineering だけが実際に作る権利を持つべきだとは言えない。

しかし、その自由によって技術的な品質まで営業担当者の責任になるわけではない。

変更がセキュリティ、権限、機密データ、アーキテクチャに影響するなら、開発の Garant はレビューを求め、修正を指示し、必要なら変更を止められなければならない。

営業担当者は自分のニーズに責任を持つ。

Engineering は、企業が技術的に維持していくものについて責任を持つ。

同じ考えは Design にも当てはまる。

PM が既存の部品を使って画面を組み立てることはできる。Design System とよく知られたパターンの範囲に収まっているなら、毎回 Design の承認を求めるのは自律性を無駄に削る。

一方、Design System を外れた変更、新しいパターン、重要で複雑な操作の設計なら、デザインの Garant が関われる必要がある。

したがって境界は「誰がどこを触ってよいか」ではない。

「**リスクや新規性がどの程度になったら、その分野の Garant が関わる必要があるか**」である。

案件の owner と分野の Garant は同じではない

これは新しい組織を考えるうえで、特に重要な区別の一つだと思う。

owner は一つの案件を担う。

問題を理解し、仕事を進め、現場の判断を下し、必要な調整を行い、結果を確認し、その案件を最後までやり遂げる責任を持つ。

一方、**Garant** は分野を横断する、より広い責任を負う。

特定の案件の担当者である必要はない。その分野の整合性、品質、リスクの一定の水準を守る役割だ。

だから Product Engineer が機能の owner でも、企業全体のアーキテクチャの Garant になるわけではない。

PM がリリースの owner でも、Design 全体の Garant にはならない。

営業担当者が業務ツールの owner でも、社内アプリのセキュリティの Garant にはならない。

この区別によって、一人にすべての分野の専門知識を求めることなく、成果に対する責任を明確にできる。

AMPECO の例にも、この動きの一形態が見える。Product Owner とエンジニアが「何を」「なぜ」をそろえ、分析、開発、テストの多くをエージェントが担当する。それでも最終的な検証の責任は人間が持つ。[6]

AI によって実装に必要な人数は減るかもしれないが、**誰が責任を負うのか**を明確にする必要は、むしろ高まる。

職種という単位では、もう捉えきれないのかもしれない

この変化は、さらに大きな問いを生む。企業を主として職種名で組織する考え方は、今でも適切なのだろうか。

Product Owner をめぐる議論は示唆的だ。私は Malorean で、PO とはバックログ管理によって正当化される職種ではなく、意思決定と責任を担う**機能・役目**だと主張した。[7]

AI の時代、この区別は特に役立つ。

実行する作業の一覧で職種を定義すると、AI によってその定義は不安定になる。多くの作業が、さまざまな経歴の人にできるようになるからだ。

しかし、何に責任を負うかで定義した役割は、ずっと安定している。

PM は仕様書を書く時間が減っても、プロダクト上の選択の品質に責任を持ち続けられる。

開発者は手作業で書くコードが減っても、技術的品質に責任を持ち続けられる。

デザイナーが画面上の要素を動かす唯一の人でなくなっても、体験の一貫性を守る Garant であり続けられる。

Product、Design、開発の境界が柔軟な Cursor でも、プロダクト責任者は、作る価値のあるものを選び、方向性から外れないために十分な指針を示すことが大切だと述べている。[8]

つまり、職種が変わるのは責任をすべて失うからではなく、一部の実行方法を独占しなくなるからだ。

Garant を新しい官僚制度にしてはならない

ただし、明らかな落とし穴がある。

PM がコードを触るたびに Engineering の承認が必要で、開発者が現場で判断するたびに PM へ戻し、デザイナー以外が作った画面は必ず Design の会議で承認を待つなら、何も解決していない。

以前の必須の引き継ぎを、新しい名前で作り直しただけだ。

Garant は常に必要な承認手順になってはいけない。

そもそも Garant を置くのは、自律性を広げるためだ。既知のルールの中なら、誰もが毎回許可を求めずに行動できる。組織は責任の所在と、いつ担当者に判断を求めるかを知っているからだ。

現実には、このモデルも完璧には動かない。

Garant ならそのままでは承認しなかったような変更が本番に入ることもある。現場の判断が気づかれないまま通ることもある。想定以上の自由を使うチームもあれば、慎重になりすぎて早い段階で上位判断を求める人もいる。

それだけでモデルの失敗とは限らない。

本当に自律的な組織は、一定の決定が事前の点検なしに行われることを受け入れる。そうでなければ、自律的とは呼べない。

目指すべきなのは、あらゆる逸脱をゼロにすることではない。

どの逸脱を許容し、どれを仕組みで検知し、どれには高くつく前に Garant が介入すべきかを決めることだ。

ここで問いが変わる。

案件を担う人の責任と、分野を守る Garant の責任を区別した。

次に答えるべきなのは、

いつ owner に一人で判断させ、いつ Garant を関与させるべきか。

Malorean Corp.でも、主体性と責任の境界が摩擦を生み始めている。次の物語でそれを見てから、歯止めとなる仕組みと、上位判断を求めるルールを検討しよう。

参考文献

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

- URL：https://www.figma.com/blog/2025-shifting-roles-report/
- 現代のプロダクト開発チームで職種の境界が曖昧になり、責任範囲が広がっていることを調べる。

[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.en.html
- AI によって行動できる人が増えても、Product、Design、Engineering の責任は消えないという著者の考察。

[3] Designers' workflow for shipping code – Atlassian

- URL： https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code
- デザイナーが製品コードを変更しても、技術チームが品質、性能、本番導入について検証する例。

[4] Product Engineer – Linear Careers

- URL： https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- 問題の定義から機能提供までの Product Engineer の責務を示す求人情報。

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

- URL：https://malorean.net/articles/2026-06-03-la-qualite-appartient-a-ceux-qui-livrent.en.html
- リリース後の品質の責任を、別の修正担当チームへ一方的に渡すべきではないという主張。

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

- URL： https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- 実装を多くのAIエージェントに任せながら、方向性と最終的な検証の責任を人が維持する事例。

[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.en.html
- バックログ等の成果物を管理する職種と、意思決定・オーナーシップを担う役割を区別する。

[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
- 職種間の境界を柔軟にしながら、プロダクトの判断、目標、制約を重視する組織例。

7 – Malorean Corp. – Alice が書き残してこなかった知識

木曜日の朝、Lucas はその週三度目となる電話を Alice にかけてきた。

「二分だけください」と約束した。

Alice は、この言葉を警戒するようになっていた。Lucas は長い会議を求めることはない。具体的な質問を持ってきて、二人で話すと、すぐ仕事へ戻っていく。だからこそ、割り込みを断るのが難しかった。

「どうぞ」

「ロット追跡の件です。顧客が仕入先で検索したいと言っています。デモに追加できます。でも、その仕入先から内部番号が同じ複数のロットが届いていたら、まとめて表示しますか。それとも選んでもらいますか？」

Alice は数秒考えた。

「一致するロットが複数あるのに、一つを特定できたと思わせるのは絶対に避けましょう。結果を分けて表示して。顧客がどう区別しているかは一緒に確認できるわ」

「僕もそう思いました。ただ自信がなくて」

「じゃあ、なぜ電話してきたの？」

Lucas は少し困ったように笑った。

「Julie のモックアップの件があってから、プロダクトのルールに関わりそうなことは、先に確認したほうがいいと思って」

先週 Alice は、見込み客の要望をそのまま標準機能に変えないように言った。だが Lucas が、少しでも迷えば彼女に聞くべきだと受け止めたのは残念だった。

「分かった。でも今回は、私に聞かなくても決められたわよ」

「そう言ってもらえると安心します」

電話を切った。七分かかっていた。Alice は書きかけの文書へ戻り、最後の一文を読み直したが、次に何を書こうとしていたのか思い出せなくなった。

十分後、Lucas からメッセージが届いた。

もう一つ。過去の製造記録でロット情報がない場合、「不明」と表示するか、列自体を隠すか、どちらがいいですか？

Alice は返信を書き始めたが、手を止めた。

質問は違っても、考え方は同じだった。ERP が持っていない情報を、確実なもののように見せてはならない。二つのケースを個別に判断する代わりに、その原則を Lucas へ伝えればよかったのだ。

それでも返信した。

情報が足りないことを明確に表示して。問題を隠さないこと。

Lucas から「いいね」の反応が返ってきた。

Alice は予定表を見た。サポートとの話し合いが延期になっていた。午後には顧客との面談がある。その合間に、複数拠点を持つ見込み客向けに Julie が作った提案を読み直さなければならなかった。

Lucas と直接話せること自体は好きだった。彼に案件を託したのも、まさにそれが目的だった。ところが Lucas の仕事が速いために、新しい可能性が現れるたびに新しい質問が生まれた。相談が増えるほど、彼女が最も力を発揮できる仕事、顧客を理解し製品の方向性を考える時間が減っていく。

Lucas も時間を失っていた。Alice の返事を待ってから、仕事に戻ることもあった。

嫌な考えが頭をよぎった。このままなら、結局すべての決定をチケットに戻すことになるのではないか。そのほうが管理は簡単だ。Alice が仕様を書き、Lucas が実装し、どこまで自律的に判断してよいか誰も悩まずに済む。

それは、二人が乗り越えようとしていた働き方そのものだった。

昼、Alice は Bob とビデオ会議で話した。二人は議題を決めずに近況を持ち寄る習慣があった。

「Lucas とはどう？」と Bob が聞いた。

「ある意味では、うまくいきすぎているわ」

「褒めているようには聞こえないね」

「すごく速く進むし、いろんなことに気づくし、顧客とも話してる。でも、自分で決められることまで、何度も私に電話してくるの」

「それで、答えてあげてる？」

「当然でしょう」

「じゃあ、電話するのが正解だよ」

Alice は目をむいた。

「ありがとう、Bob」

「半分は冗談だけどさ。間違える危険を取るより、五分で君から答えがもらえるなら、僕だってそうするよ」

「でも、それじゃいつまでたっても、このやり方で仕事ができない」

Bob は例を聞いた。Alice は朝の二つの質問、Julie との話し合い、それから Lucas が最近迷っていた、ほかの顧客にはほとんど使われない設定を一社のために追加すべきかという問いを話した。

「気づく？」と Alice。「いつも同じ問いの応用なのよ」

「で、君の答えは？」

「場合によるわ」

Bob は笑った。

「ほら、それだよ。だからみんな君に電話するしかないんだ」

Alice は反論しかけたが、確かにそうだった。

「適当に『場合による』と言ってるわけじゃないの」と言い直した。「繰り返し起きる問題なのか、優先顧客に役立つのか、ほかの顧客まで製品が複雑にならないか、保守できるのか……」

途中で言葉を止めた。

「ほらね」と Bob。「今、判断するためのルールを一部説明したじゃないか」

「ルールっていうより、考え方よ」

「なおさらいい。Lucas に伝えるべきなのは、それかもしれない」

Alice は黙った。

何年もチケットを書いてきた。期待する動作、例外ケース、受け入れ条件は記述した。しかし、複数の選択肢から一つを選ぶときの考え方は、ほとんど書かなかった。その判断の根拠は彼女の頭の中に残るか、リリース後に会議と一緒に消えてしまっていた。

「そんな知識が何百もあるかもしれない」と Alice。

「きっとあるよ。僕にもね」

「書く理由がなかったもの」

「理由はあったよ。ただ時間がなかった。それに『良いアイデアでも断る理由を説明する』なんてタイトルのチケットは誰も作らなかったから」

二人は笑った。

電話のあと、Alice は新しい文書を開いた。*プロダクトの意思決定原則*と書いた。

最初の一文を追加しては消し、別の書き方を試した。十五分後、そこには社内規程のような段落があった。内容は正しかった。しかし Bob に説明したときの言葉とはまるで違った。

Alice は文書を閉じ、ChatGPT を開いた。

8

「私たちの製品について、頭の中にあることを言葉にするのを手伝って」と音声入力した。「机上の手順書はいらない。質問して、実例を思い出させて、本当のルールと私個人の習慣を分けるのを手伝って」

最初の返答は、最近の意思決定から始めることを提案した。

Alice は話し始めた。

Julie の見込み客について。魅力的なモックアップと、ある顧客だけの要望を全顧客への約束に変えてしまう危険について。大口顧客が大きな売上をもたらしても、それだけで製品にとって正しい方向とは限らないことについて。

「待って。理由を説明しないと」と声に出した。「うちの ERP はいろいろな会社に使われている。特別なルールを全部ソフトウェアの中心に入れたら、全員の仕事が複雑になる。でも、特別な要望がもっと普遍的な問題を教えてくれることもある。だから何でも断ればいいわけでもない」

ChatGPT は、その違いをどう判断するのかと尋ねた。

Alice は、サポートとの対話、複数顧客で見られたケース、営業からの要望、導入時の制約について説明した。記録の残っていない古い議論も思い出した。ある案をなぜ見送ったか、なぜ新機能より設定項目を選んだか、給与計算の例外をなぜ画面の好みと同じように扱えないかを話した。

時には、自分がそれまで明確に考えたことのない問いに答えていることに気づいた。

AI は話を言い換え、整理し、原則を提案した。Alice は修正した。

「違う。『個別開発は絶対にしない』じゃない。条件によってはできるけれど、何を犠牲にするか意識して決める必要があるの。それに、営業上の仮説をプロダクトの決定として扱ってはいけない」

しばらくすると、ChatGPT はロット追跡機能によって過去のすべてのロットの履歴を復元できる、と書いた。

「そんなわけないでしょう」と Alice は画面に向かって言った。「記録していないデータを魔法のように復元することはできない。それは約束すべき能力ではなく、明示すべき制約よ」

該当箇所を修正させた。決定済みのこと、可能性が高いと考えていること、サポートや Lucas や顧客に確かめるべきことを、文書で区別するよう求めた。実際の顧客アカウントの機密情報を貼り付けることは避け、事例を言い換えた。

一時間が過ぎ、二時間が過ぎた。

文書には、Alice が長年知っていたことが書き込まれていった。Malorean が優先的に支えたい顧客、何度も起きる問題、受け入れてきた妥協、してはならない約束、古いデータの限界、試作品がそのまま販売できる機能ではない理由。

すでにある情報へのリンクも追加した。サポートのフィードバック、まだ追跡できる意思決定、商談メモ、製品のドキュメント。根拠が欠けている箇所には、そのことを書くようにした。

三時間目が終わるころ、ChatGPT は要約を提案した。

Alice は読み直した。長すぎるページも、一般原則と具体例が混ざった箇所もあった。断定しすぎている表現をもう一度直し、日付と文書の対象範囲を書き足した。

それから全体を PDF に書き出した。

ページをスクロールし、その多さに驚いた。

三時間前は、Lucas がいつ自分に聞かずに決められるか、説明しようとしていただけだった。ところが今、チームが知っていても一度も共有しなかった製品知識が、不完全ながら驚くほど豊かにまとめられていた。

最終的な真実というわけではない。確認や更新が必要な箇所もある。

PDF が顧客やデータ、担当者の判断に取って代わるわけでもない。

それでも、次の人がゼロから始めなくて済むようにはなった。

翌朝、Lucas はファイルを開いた。

「これ、全部僕のために書いてくれたんですか？」

「一人で書いたわけじゃないの」と Alice。「ChatGPT と三時間話したの。質問してもらって、答えを整理して、抜けていることを見つけてもらった。私は読み直して直した。でも内容のほとんどは、前から知っていたことよ」

「じゃあ、どうして今まで見たことがなかったんだろう？」

Alice は少し照れたように笑った。

「チケットに書く内容じゃなかったから」

Lucas は目次を見た。原則、意思決定の例、業務上の制約、顧客情報、資料へのリンク、未解決の問いが並んでいる。

「ロット追跡を開発するとき、この文書を僕の AI に渡していいですか？」

「ええ、許可された環境でなら。でも、文脈を理解するための情報として使って。全部正しい証拠だと思わせないこと。大切なルールなら、元の資料や責任を持つ人に確認するのよ」

「ものすごく時間が節約できます。二つの案で迷って AI に相談することがあるけど、AI が知っているのは僕の説明だけですから。三年前の決定を知らずに、もっともらしい答えを出すこともあるんです」

Alice にも分かった。単独で見れば妥当な答えが、彼らの ERP にとってはまったく筋違いになることがある。

Lucas はある箇所で止まった。

「ここに、結果が完全だと誤解されるおそれがあるときは、欠けている業務情報を隠してはいけないと書いてあります」

「昨日、そう言ったでしょう」

「はい。でも昨日この一文を持っていたら、電話しなくて済んだのに」

Lucas は読み進め、顔を上げた。

「なるほど。でも、まだ問題があります」

Alice は少しがっかりした。

「何？」

「Alice がどう考えるかは、前よりずっと分かります。でも、僕がどこから一人で決めていいのかは、まだ分かりません」

Alice はカップを置いた。

Lucas は、文書の最も大きな欠落を見つけたのだ。

二人は再び話し始めた。Alice は空白のページを共有した。

「簡単に考えましょう。私に電話する前に三つ質問するの。求める結果を理解しているか。決定が既知のルールの範囲にあるか。そして間違えたら、どんな問題が起こるか」

「『ルールの範囲』を定義しないといけませんね」

「そのとおり」

二人は三つの段階を書いた。

緑：判断がその場限りで、元に戻せて、既知のルールに沿い、顧客への新たな約束を含まないなら、Lucas が一人で決める。理由を説明し、結果を確かめることができる。

オレンジ：新しい状況、目標の衝突、製品の動作について重大な疑いがあれば、Alice に相談する。技術的な危険があれば、ERP の履歴について Thomas に聞くように、専門家にも相談する。

赤：製品の基本ルールを変える、Malorean として商業的な約束をする、機密データに触れる、重大で元に戻しにくい危険を取る、といったことは一人で決めない。関係する責任者との明確な判断が必要で、ときには Hubert も加わる。

Lucas は一覧を読み返した。

「誤解されないようにデモの結果表示を変えるなら、緑ですね」

「そうね」

「全顧客に関わるかもしれない動作を追加するなら、Alice に相談する」

「話し合いましょう。特にニーズがよく分かっていない場合は」

「履歴データを変更してロットを記録したいなら？」

「私一人で許可できる話じゃない。技術的な分析と適切なチェックが必要ね。あなたにずいぶん念を押されたし」

Lucas は笑った。

「覚えているか確認したかったんです」

二人はルールを一つ付け加えた。Alice から返事がないからといって、オレンジや赤の決定が緑に変わることはない。一方、緑の判断すべてに承認を求める必要もない。

「色を間違えたら？」と Lucas。

「教えて。ルールを調整するわ。私の考えを当てられなかったと責め続けるより、仕組みを良くしたいもの」

Lucas はうなずいた。

「AI に自分の判断をこの一覧と照らし合わせて、理由と足りない情報を示させることもできますね」

「ええ。ただし、そのリスクを取っていいかを AI に決めさせないこと」

Alice はこの一覧を参照文書に加えた。単に情報を伝えることと、本当に判断できるようにすることの違いが、少しずつ見えてきた。

ここで終わってもよかった。しかし Lucas はロット追跡のデモへ話を戻した。

「一時間も僕の自律性について話しました。今度は案件そのものについて質問します。何ができれば成功と言えるんですか？」

Alice はほとんど反射的に答えた。

「顧客が、あるロットを使って作られた製品を探せれば」

「そうです。でもどう確認しますか？　どのくらい信頼できなければならないんですか？」

今度は、すぐ答えようとはしなかった。

二人は何日も画面、ルール、データ構造を調べてきた。見込みのあるデモもあった。それなのに、結果をどう判定するかを正確には誰も書いていなかった。

Alice は、仕入先からの警告のあと半日以上もファイルを照合していた品質管理責任者を思い出した。

「そこから始めましょう。警告されたロット番号を使って、手作業の調査を繰り返さずに対象の製造記録を探せるか。どれだけ時間がかかるか。結果は完全か。情報がないとき、それを明確に表示できるか」

「本人と一緒に試せますね。デモが気に入ったか聞くだけじゃなくて」

「そういうこと」

Alice は新しいページを開いた。

「本当は、作り始める前にここから考えるべきだったわ」

Lucas は声を出して笑った。

「二週間前に言われたら、また僕にチケットを作ろうとしているって思ったでしょうね」

「かもしれない。でも今書いていることを見て」

二人は、観察した問題、対象となる利用者、期待する結果、測り方、データの限界、対象外の範囲、承認が必要な決定を書いた。一ページ、それだけだった。

Lucas は*Mini PRD*と呼ぶことを提案した。

「どの画面を作るか指示されているわけじゃないですね」と言った。「この仕事が役に立ったと言える条件を、二人で確認しているんだ」

「越えてはいけない制約もね」

Alice は考えを声にした。

「次の案件でも、ここから始めるべきね。何を開発するかを尋ねる前に、どんな結果がほしいか、どう確認するかを考える。そのあとで初めて範囲を決め、何を作る価値があるか選ぶ」

「僕が必要な情報を使ってこの文書を書ければ、Alice がバックログを作るのを待たずに始められますね」

「課題の整理を提案できるようになる、ということよ。どんな約束でも見ずに承認するわけではない。でも、それが目的ね」

二人は、まだ判断できない問いを未解決のまま残して、ロット追跡の Mini PRD を整えた。顧客に確認すべきケースもあれば、技術上の制約を確かめる必要もあった。

その文書は、確実だという錯覚を与えなかった。何を学ぼうとしているのかを示していた。

Alice は、機能のチェックリストよりずっと役に立つと思った。

翌月曜日、Alice と Hubert の定例会議はいつもと違った。

Alice は通常の進捗表も、Lucas が作った機能の一覧も見せなかった。

判断の三段階の一覧と Mini PRD を見せた。

Hubert は冒頭を読んだ。

「Lucas はずっと速く作れるようになったのに、こんなに文書を書く時間を使ったの？」

Alice には、その指摘が分かった。数日前は自分も同じことを考えていたからだ。

「むしろ、今まで必要なことを書いていなかったんです。何を作るかは書いても、どう選ぶかの情報はほとんど残していませんでした。その結果、Lucas が常に私へ判断を求めた。この文書があれば、私を待たずに決められることが増えます」

「制約を守っているか、どう分かるの？」

「一人で決めていい場合と、相談すべき場合を決めました。そのほかのことを全部点検するつもりはありません。結果を確認して、必要ならルールを直します」

Hubert は、古いデータについて書かれた行に目を止めた。

「そもそも記録されていないものは、追跡できると保証できない」

「ええ」と Alice。「文書にもそう書きました。顧客に明確に伝えないといけません」

Hubert はうなずいた。

「営業が逆のことを約束する前に、それを知っておきたいね」

「そのためにも、この情報は Lucas だけのものにしないほうがいいです」と Alice。「Julie も、分かっていることとまだ決めていないことを確認できます。ただ、更新し続けて、どの部分が信頼できる判断基準なのかを全員が知る必要があります」

二人は、書き換えられる作業版を残し、PDF は日付付きの記録とすることにした。重要な決定は理由とともに記録する。繰り返し使う原則は、適切であれば、エージェントへの指示やツールによる自動チェックにできる。ただし、技術的なテストや本番導入前の必要な承認は残る。

「半年後に、誰も支持していない古いルールを AI が適用していたなんて知りたくない」と Hubert。

「私もです」と Alice。

そこで初めて、Hubert は Mini PRD に強い関心を示した。

「顧客が本当に自分のロットを追跡できるか、確認するんだね？」

「はい。一緒に試してもらいます。結果が信頼できなければ、画面が何枚完成していても意味はありません」

Hubert は文書を閉じた。

「いいね。何を納品したかだけでなく、何を学んだかを知りたい」

水曜日、Alice はメッセージ欄に妙なことを見つけた。

朝から Lucas のメッセージが一通もない。

プロジェクトのチャンネルを開くと、活動は続いていた。Lucas はデモ環境で複数の結果表示を試していた。一つの案については、同じ参照番号のロットを分けて表示した理由を書いていた。存在しない関連があると思わせないためだ。プロダクトの参照文書にある原則を引用し、使ったデータを示し、顧客に画面の意味が伝わるか確かめることを提案していた。

許可は求めていなかった。

Alice は説明を読み、仕事へ戻った。付け加えることはなかった。

夕方、Lucas から電話があった。

「オレンジなんですけど、赤に近いかもしれません」

「聞かせて」

「顧客が、古い製造記録からも検索したいそうです。履歴の一部には原材料と完成品の確かな関連がありません。分かっていることは表示できます。でも『すべて検索できます』と見せてしまうと、結果が出ないことを『対象製品はない』という意味に受け取られるおそれがあります」

Alice は姿勢を正した。

「そのとおり。それは曖昧にできないわ」

「警告を表示して、データが利用できる期間とそうでない期間を分けられます。ただ顧客に何を約束していいか、一緒に決めたいです」

「これは一緒に決めましょう。サポートにも、この制約がどう伝わるか聞く必要があるわ」

Lucas は画面を共有した。二人は問題に取り組み始めた。ボタンを承認する話でも、Alice がすでに答えを知っている質問でもなかった。顧客がソフトウェアに寄せる信頼そのものを左右する決定だった。

話しているあいだ、Alice は一度も、この会議に何分かかっているか気にしなかった。

電話を切る前に Lucas はためらった。

「Alice、あの PDF では、僕の質問が全部解決したわけではありません」

「それでいいのよ」

「いいんですか？」

「もし全部の答えを書けたなら、あなたが新しく見つけることは何もなくなってしまうもの」

Lucas は笑った。

「なるほど。でも、いつ相談すればいいかは、前より分かるようになりました」

電話のあと、Alice は予定表を見た。次の顧客面談まで一時間空いていた。

まるまる一時間だ。

この実験が大きな組織でもうまくいくかは分からない。今後、どれほど失敗するかも分からない。ただ、Lucas がいくつかの判断を一人で進め、本当に彼女の判断が必要なときに戻ってきたことだけは確かだった。

Alice は共有文書に、過去データをどう示すかという新しい問いを加えた。Lucas に何を作るべきか命じるチケットではない。関係者と検討すべき決定だ。

仕事が簡単になったわけではない。

注意を向けるべき場所が、ようやく変わり始めていた。

8 – 自律性を高めるほど、明確なルールが必要になる

企業は、専門家を待たずに、より多くの人が行動できるようにしようと決めることができる。

しかし、これまで見てきた職種の境界の変化と責任の分担は、新たな調整上の問題を生む。同じ案件に複数の人が関われるようになっても、判断の基準が必ずしも共有されているとは限らない。

一つひとつの行動に毎回人間の承認が必要なら、自律性は名ばかりだ。

職種間の必須の引き継ぎを、別の場所へ移しただけになる。

本当に重要なのは、整合性、安全性、品質、製品の方向性を損なわずに、より多くの決定を現場で下せる仕組みを作ることだ。

そこには重要な逆説がある。

自律性を広く分散させたいほど、判断のためのルールを明確にする必要がある。

自律性は、ルールがないことから生まれるのではない。

目的、限界、リスク、責任が十分に明確だからこそ、毎回許可を求めずに行動できるのだ。

すべてを承認しなければならないなら、自律性はない

一つの案件を任された Product Engineer を想像してみよう。

利用者と話し、解決策を探り、試作品を作り、開発し、リリースを準備する。

書類の上では自律的だ。

しかし、製品についての細かな判断はすべて PM に、画面の細部はすべてデザイナーに、技術的な選択はすべて Tech Lead に確認しなければならないなら、責任のあり方は実際には変わっていない。一人の人間が、より多くの承認手順を通らされているだけだ。

ほかの職種も同じだ。誰かに作る力を与えても、あらゆる行為に昔と同じ手続きと承認が必要なら、得られるものは少ない。

問いは、単に**誰が行動できるのか**ではない。

その人がどんな条件の下で、一人で決められるのかを明確にすることだ。

Cursor では、製品のビジョンやロードマップから外れずに作り手が動けるよう、目標と歯止めとなる条件を十分に明確にするという考えが示されている。[1]

こうして、詳細な指示の一部を、判断のための共通の枠組みに置き換えることができる。

詳細なチケットの一部を、判断の基準に置き換える

Linear の事例から、完成形まで指示されたチケットが主体性を狭める場合があることを見た。[2] では実際に、追加の仕様書を待たずに判断するためには、どんな情報が必要だろうか。

これはチケットをなくす話ではない。私は Malorean で、バックログは行動と調整のための道具であって、製品についての記憶をすべて保存する場所ではないと主張した。[3]

自律性のためのルールを明確にするなら、解決策を先に固定するのでなく、現場の判断を導く情報を示すことになる。

例えば、次のようなことだ。

- 優先する顧客や理想的な顧客像（ICP）
- 解決すべき問題
- 期待する結果
- 対象範囲の限界
- 重要な制約
- 求める品質の水準
- 企業として、あえて取り組まないと決めたこと
- それを超えたら上位判断を求めなければならない基準

Ramp は、この考え方の具体的な例を示している。中心となる ICP と問題、特に優れた形で解決したいことを明確にすれば、製品を別の方向へ引きずる収益性の高い機会であっても、断る判断ができる。[4]

Figma も、素早くゼロから一を作るだけでは十分でないと指摘する。速い試作を良い製品に変えるには、実際の利用場面、例外、品質の水準が引き続き重要だ。[5]

この一覧は、何をすべきかだけでなく、**守らなければならない限界を見えるようにする**という組織上の選択を示している。ここで万能の方法として提案しているわけではない。

文脈の共有が、組織の基盤になる

現場で良い判断をするには、問題を理解するための情報にアクセスできなければならない。

これは、AI エージェントの時代における、最も重要な組織変化の一つかもしれない。

少数の専門家の頭の中に大切な情報を閉じ込めたまま、より多くの人に判断を任せることはできない。Linear は、顧客の声、戦略の方向性、決定、コードを、人間にもエージェントにも利用できるようにすべきだと述べている。[2]

私は Malorean で、PM を**コンテクストの設計者**として捉えた。インタビュー、営業での反論、サポートの情報、決定、データ、資料、小さな兆候といった散在する知識を、組織が再利用できる資産に変える役割だ。[6]

この考え方は、ドキュメントの意味を大きく変える。

目的は、ページを増やすことではない。

誰かが一からすべての考え方を再構築しなくても判断できるよう、必要な情報を使える形にすることだ。

良い背景情報があれば、次のことが分かる。

- 顧客が実際に何を言ったか
- すでに検証した仮説は何か

- どんな制約が重要か
- どんな大きな決定がなされたか
- これまでどんな妥協を受け入れたか
- 結果を判断するための指標は何か

自律性が高まるほど、こうした情報は信頼でき、共有されていなければならない。

そうでなければ、行動する力だけを分配し、理解する力は分配しないことになる。

「なぜそう決めたか」を忘れない

判断が現場へ移ると、別の問題も生まれる。

決定が違う人によって、時には以前の議論に参加していなかったエージェントを使って行われるようになる。

理由の記録がない決定は、やがて再び議論の対象になる。

結論は覚えていても、その背景は忘れてしまう。

ルールの存在は知っていても、なぜ選んだのか分からなくなる。

新しい顧客が現れ、マネージャーが交代し、チームが変わる。すると、すでに決着した議論が、以前より少ない情報で繰り返される。

これを防ぐための道具として私が考えているのが、**Product Decision Record（PDR）**だ。[7]

PDR は細かな判断をすべて記録するものではない。

複数のチーム、製品、将来の場面で判断を導くような、重要なプロダクト上の決定を残す。

主に、次の情報を保存する。

- 当時の背景
- 検討した代替案
- 選んだ決定
- そう選んだ理由
- 良い面と悪い面の影響

- 受け入れたリスク
- 決定を見直すべき条件
- 判断した人と相談を受けた人

事務作業を増やすためではない。

良い PDR は、同じ話を五十回やり直すのを避けるためにある。

ここに重要な区別がある。

コードは何を作ったかを残す。意思決定の記録は、なぜその重要なルールを選んだかを残さなければならない。

長く続く自律性には、その両方が必要だ。

歯止めとなるルールを、仕組みに組み込む

良い背景情報があっても、全社員に会社のルールをすべて覚えさせるのは非効率的だ。

そこで、繰り返し使うルールを人間の承認手続きからツールへ移す企業がある。実際に見られる取り組みではあるが、自動化だけで品質が保証されるという意味ではない。

Meta は Diff Risk Score を使って、変更が障害を起こす確率を見積もり、リスクに応じて扱いを変える仕組みに役立てている。[8]

Cloudflare は明文化した Engineering の標準を自動チェックに変えている。必須ルールに違反する変更は、修正するか明確な例外承認を得るまで、承認できない場合がある。[9]

この動きは専門家の役割を変える。

技術の Garant は、同じルールを五十件のプルリクエストで繰り返し説明する必要がない。自動で検証できるなら、その仕組みに任せられる。

Design System によって正しい選択を簡単にし、間違った選択を見つけやすくできるなら、デザインの Garant はすべてのボタンを承認する必要がない。

重要な決定、対象顧客、目標が共有されていれば、プロダクトの Garant も毎週同じ制約を説明し直す必要がない。

判断のためのルールそのものが、組織の基盤になる。

テスト、CI、Design System、標準、セキュリティルール、テンプレート、エージェント向け指示、指標、共有された背景情報が、以前は人が担っていたチェックの一部を引き受ける。

人間の専門知識は、例外的な問題へ向けられるようになる。

すべての決定に同じチェックは必要ない

ここで重要になるのが、リスクという考え方だ。

文章の翻訳と権限管理システムの変更を、同じ承認手順で扱うべきではない。

小さく元に戻せる変更と、データ移行や会社のポジショニングを変える決定には、同じ水準の確認は必要ない。

「承認が必要か」という二択ではなく、「**このリスクにはどの程度のチェックが妥当か**」と問うことができる。

次の三段階は**本書が提案する判断のための整理**であり、紹介した企業すべてに共通する実務を示したものではない。

現場の自律的な判断

決定は既知の目標と標準の範囲に収まる。

元に戻せて、想定される影響が限られ、重要なルールを変えない。

案件の owner が決める。

対象を絞った確認

その分野で新しいこと、例外、重大なリスクが生じる。

関係する Garant が関わる。開発、デザイン、プロダクトなど、必要な分野の責任者だ。

案件そのものを奪うとは限らない。足りない専門判断を補う。

上位へのエスカレーション

決定によって、仕組みそのものの限界を変える。

技術面なら、基本的なアーキテクチャ、セキュリティ、機密データ、重大なデータ移行、規制リスク。

デザイン面なら、重要な新しい操作、Design System からの逸脱、ほかの利用手順に大きな影響を与える変更。

プロダクト面なら、対象顧客、価値提案、価格、巨額の投資、ロードマップ、戦略上の大きな判断だ。

この場合、owner は既存のルールの**中で**決めているのではない。

ルールそのものを変えようと提案している。

本書で考えるモデルでは、Garant や、さらに上位の意思決定者が関わるべき状況になる。

限界が不明確だと、逆方向の二つの危険が生じる

意思決定の限界が明確でない組織は、反対方向の二つのリスクにさらされる。

第一は官僚化だ。

すべての決定が上へ回される。専門家が順番待ちの窓口になり、責任者は些細な判断まで承認しなければならない。自律性は消える。

第二は責任の曖昧化だ。

現場の全員が、一つの判断がいつ現場だけの問題でなくなるか分からないまま決める。矛盾した選択が積み重なり、顧客や企業に見える問題として現れる。

Malorean では、こうした境界を明確にするための組織の道具をいくつか紹介した。共通言語、one-pager、指標の階層、意思決定の記録、エスカレーションのルールだ。[10]

価値があるのは、文書を増やすからではない。

次の問いを明確にせざるを得なくするからだ。

- 誰が決めるのか
- 何について決めるのか
- いつ決めるのか
- 何を現場で決めてよいのか
- 何を上へ伝えるべきか

- どんな条件で決定を見直せるのか

二つの危険から、一つの組織上の仮説が生まれる。承認を増やしたり減らしたりするより、**決定が行われる場所を適切にすること**が重要かもしれない。

決めるとは、何かを諦めることでもある

歯止めになるルールは、技術面だけでは足りない。

本当の選択をする能力も、その重要な一部だ。

組織では、全員が合意しているように見せるために、曖昧な決定が行われがちだ。各自が自分の優先事項を読み込み、実行段階で初めて、両立しない条件だったと分かる。

私はこれを「**曖昧な決定**」と呼んでいる。[11]

自律性の高い仕組みでは、とくに危険だ。

目標が曖昧なら、各チームはその場では妥当でも、全体としては矛盾する判断を下し得る。

だから自律性を高めるには、何を諦めると決めたかも見えるようにしなければならない。

どの顧客を優先するのか。

今はどの顧客層を対象にしないのか。

譲れない品質の水準は何か。

どの制約なら一時的に緩められるか。

目標が衝突したら、何を優先するか。

自律性には、すべてを規定するルール集はいらない。

現場の判断が同じ方向を向くよう、本当の選択を十分に明確にする枠組みが必要なのだ。

成果を測ることが、最後の歯止めになる

自律性は、何かをリリースした瞬間に終わるのではない。

本当に案件を担っているなら、その後に何が起きるかも見なければならない。

最初の章では、生産した量と生んだ価値を区別した。ここでは問いが少し違う。**決定した人は、その判断が適切だったかどうやって後から知るのか。**

Canva の仕組みが具体例になる。チームは実験と、その事業上の推定影響を結びつけられる。[12]

一つの案件ごとに、次のことを明確にする必要がある。

- どんな変化を期待していたか
- それを観察するための指標は何か
- 守るべき品質は何か
- 利用者は実際に何をしたか
- 続けるか、修正するか、やめるか、さらに調べるか

測定によって、責任の循環が閉じる。

測定がなければ、自律性は単に「作ってもよい」という許可になりかねない。

測定があれば、結果に責任を持つことになる。

自律性とは、マネジメントが手を引くことではない

1.4 章で紹介した Marty Cagan の自律的なプロダクトチームのモデルは、ここでマネジメントの役割を明らかにする助けになる。完成した解決策を渡さずに課題を任せるには、目標と戦略上の背景を明確にする必要がある。[13] Cagan は、技能の育成、支援、リーダーシップの重要性も強調する。チームに判断を任せることは、放任することではない。[14]

ただし、この比較には重要な限界がある。Cagan が論じるのは、補い合う技能を持つプロダクトチームの責任だ。本書で提案する Garant は、実行する人が職種の境界を越えるときに、分野の責任をどう残すかという、別の問題に対応する。

Garant は、すべての行為を承認する人ではなく、判断の枠組みを守る人になる

1.6 章では、すべての作業を自分で行わなくても、分野の整合性に責任を持つ人として Garant を紹介した。ここでその仕事は具体的になる。理解しやすいルールを定め、繰り返しのチェックを自動化し、チームが独自に決められる範囲をリスクや新規性が超えるときに介入する。

すべてを点検すれば、Garant 自身がボトルネックに戻ってしまう。誰も何にも責任を負わなくなれば、責任は曖昧になる。

だから自律性とは、すべてを承認することでも、すべての管理をなくすことでもない。責任を明確に保ちながら、より多くの決定を現場でできるようにすることだ。

しかし、この変化は Product と Engineering の職種だけにとどまらない問いを生む。

背景情報が直接共有され、判断が現場へ移り、調整の一部をエージェントが引き受けるようになるなら、**以前その仕事を担っていた中間管理職の階層は、どうなるのだろうか。**

まず Malorean Corp.の社員たちが、この問いにどう直面するか見てみよう。そのあと、企業全体への影響を検討する。

参考文献

[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
- 目標と制約を明確にすることで、役割の境界を越えても製品の方向性を維持する実践。

[2] Issue tracking is dead – Linear

- URL：https://linear.app/next
- チケットを中心とする作業の受け渡しを越え、顧客の声、戦略、決定、コードを共有する仕組みを提案する。

[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.en.html
- バックログを長期的な記憶の保管庫と混同せず、行動と調整の道具として位置づける。

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

- URL：https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- 中核顧客と重要課題に集中し、製品の方向性から外れる機会を断る必要性。

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

- URL： https://www.figma.com/blog/what-it-takes-to-build-great-products-now/
- 試作の速さだけではなく、実際の利用場面、例外、品質を重視する考察。

[6] Le PM comme architecte du Contexte – Malorean

- URL： https://malorean.net/articles/2026-06-08-le-pm-comme-architecte-du-contexte.en.html
- 分散した顧客情報、資料、決定、判断の根拠を、再利用できる文脈へ整理する PM の役割。

[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.en.html
- 重要な決定の背景、代替案、選択、結果、見直す条件を残す方法。

[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/
- ソフトウェア変更の事故リスクを推定し、確認手順を調整する仕組み。

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

- URL： https://blog.cloudflare.com/engineering-standards-enforcement/
- エンジニアリング標準を自動チェックに変え、違反をマージ前に見つける事例。

[10] Les outils de cohérence organisationnelle – Malorean

- URL： https://malorean.net/articles/2026-06-02-les-outils-de-coherence-organisationnelle.en.html
- 共通言語、目標、指標、意思決定記録、エスカレーションを通じて組織の判断を整える。

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

- URL： https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.en.html
- 諦めるものを明確にせず、見かけの合意だけを維持する曖昧な意思決定の危険性。

[12] Measuring Commercial Impact at Scale – Canva

- URL： https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- 実験による推定事業効果を、製品の判断と価値創出につなげる仕組み。

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

- URL：https://www.svpg.com/team-objectives-empowerment/
- 戦略上の背景とともに、解決すべき課題をチームに渡す考え方。

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

- URL：https://www.svpg.com/product-leadership-is-hard/
- 自律的なプロダクトチームには、放任ではなく、より高度な支援とリーダーシップが必要だと説明する。

9 – Malorean Corp. – 二つの妥当な判断

月曜日の朝、二つの開発チームの調整を担当する Vivien は、何年も繰り返してきた質問で会議を始めた。

「今週の案件がどこまで進んだか、教えてくれるかな？」

いつもの管理表はすでに開いていた。チケット、担当者、期限、障害が、長い時間をかけて磨いてきた列に整理されている。五人ずつの開発チームのことはよく知っていた。それぞれの強み、習慣、念を押したほうがよい案件、少し時間を与えたほうがよい案件も分かっていた。

誰もすぐには答えなかった。

やがて Lucas が話し始めた。

「ロット追跡の Mini PRD は金曜日に更新しました。顧客とロット番号による検索を試しました。次はデータが不完全なケースの検証です。未解決の質問は共有コンテクストに書きました」

「納期は？」と Vivien。

「過去の在庫データをどうチェックするか次第です。Thomas と確認することがあります」

Thomas はうなずいた。彼は出荷準備の流れを改善する仕事を担当し、進捗もすでにツールに記録していた。

Vivien はほかの開発者にも順に聞いた。答えは似ていた。全員が自分の問題をより深く理解し、いくつかの判断を下し、次に何をするか説明できた。仕事を分ける手助けが必要な人も何人かいたが、多くは会議を待たずに動いていた。

最後に Vivien は管理表を見た。

今聞いたことはすべて、すでにそこに書かれていた。しかも、より詳しい説明まで付いていた。

「よし、これで全部だね」

会議は十二分で終わった。

以前なら四十五分かかっていた。

喜ぶべきことなのか、Vivien にはよく分からなかった。

Vivien は Malorean で長く働いていた。開発者全員がル・アーヴルのオフィスに集まっていた時代も知っている。リリースを段取りし、遅れに対処し、プロダクト部門と優先順位を交渉し、金曜日の予定だった依頼が翌週の火曜日まで出せない理由を経営者に説明してきた。

新型コロナの流行後、会社がリモートワークへ移ったときには、自分の方法を変えた。廊下での会話だけでは仕事を回せなくなった。決定を文書に残し、障害を見えるようにし、他部署と何を決めたか各自に記録させた。

誰も全体を把握していない情報を、組織の中で行き渡らせること。それが日常の仕事になった。

Vivien はその仕事が好きだった。うまくいかなかった企画が、必要な人たちを結びつけたことで動き始める瞬間は、とくに好きだった。

しかし AI ツールがチームに入り込んでから、何かが変わっていた。開発者の実装は速くなったが、それだけではない。案件の担い方そのものが変わった人がいた。Lucas は顧客と直接話していた。Alice は、何を作るかの一覧だけでなく、決定の理由を説明するプロダクトの背景情報を共有するようになった。Bob も自分の案件で同じことを始めていた。

そして今では、以前 Vivien が手作業で作っていた要約の一部を、エージェントが準備してくれる。

仕事が消えたわけではない。ただ少しずつ自分の手を離れ、別の場所で行われるようになっていた。

そのことは、まだ誰にも話していなかった。

翌水曜日、Lucas は Alice にロット追跡デモの新しい版を見せた。

「仕入先が問題のあるロットを知らせてきた場合を試しました。分かっている製造履歴は追えます。それに、『確認中』という状態を追加しました。問題がないと確認されたわけではない、と分かるように」

Alice は画面を見た。

状況を確認する必要があることを示す表示があった。古い製造記録には、情報が一部欠けていることを知らせる別の文も出ていた。

「最初の版より誠実な表示ね」と言った。

「ええ。顧客の品質管理責任者にも、『問題が見つかっていない』と『判断に必要な情報がない』を混同しないでほしいと言われました」

Alice はうなずいた。共有した文書でも、すでに取り上げた区別だった。

「Mini PRD にも書いた？」

「はい。テストケースも追加しました」

「それなら続けて」

Alice は表示の場所や文言を一つずつ決める必要がなかった。Lucas には目指す結果が分かっていた。表示方法を選び、顧客に確かめる準備をしていたのだ。

少し後、Thomas は Vivien に出荷機能の改善を見せた。

別の問題だった。いくつかの顧客では、明示的な禁止がないにもかかわらず、中途半端な状態や誤って入力された状態のために注文が止まることがあった。物流担当者は、特定された危険がない書類を修正するのに時間を失っていた。

そこで Thomas は出荷準備のルールを明確にした。「ブロック」とされたロットは出荷できない。そのほかの状態は見えるようにしたうえで、既存のチェックを通し、通常の手順で扱うことにした。

「ほかの機能の動きは変えていないね？」と Vivien が聞いた。

「既存のケースでは変わっていません。回帰テストも追加しました。ブロックのルールには触れていません」

Vivien は Thomas の慎重さを疑わなかった。Thomas は、古いドキュメントにも残っていない例外まで知っていることがあった。

「いいね。この先へ進む前に、レビューしてもらって」

Thomas はうなずいた。

二つの変更は、まだ一つの利用場面として組み合わせて試されていなかった。

問題が現れたのは金曜日、機能を統合したテスト環境でのことだった。

サポート担当者が一連の手順を用意していた。原材料の入荷、製造、仕入先からの警告、対象製品の特定、そして出荷準備だ。

Lucas はビデオ会議でデモを見ていた。Thomas も参加していた。Alice は、その仕組みを作っていない人がどう理解するかを見るため、招待してもらっていた。

最初の数分は順調だった。

ロット番号が見つかった。製造記録が表示された。「確認中」の表示も出ていた。

そこからサポート担当者が出荷機能へ移った。

納品書は準備中の状態になっていた。しかし、作業を進めることを止める警告はなかった。

「待って」と Alice が言った。「同じロットの話よね？」

Lucas は画面に顔を近づけた。

「はい。どうしてですか？」

「上では確認中になっているのに、ここでは何も問題がないかのように進められるわ」

Thomas は注文の詳細を開いた。

「ブロックされていません。確認中なんです。同じ状態ではありません」

「僕は」と Lucas が答えた。「確認中というのは、誰かが判断するまで、承認されたものとして扱ってはいけないという意味にしていました」

Thomas はカメラのほうを向いた。

「なら、どうして『ブロック』にしなかった？」

「一律に禁止したくなかったからです。調査は必要でも、業務を全部止めなくてよい顧客もいます。本当にどこまで問題なのかは、顧客の品質管理責任者が判断すべきです」

「僕の機能には」と Thomas が言い返した。「履歴データが欠けている古いロットもある。明確に承認されたものしか出荷できないようにしたら、いつまでも動けない注文が出てくる」

Alice は二つの画面を見比べた。

Lucas の理由も分かった。Thomas の理由も分かった。

問題は、利用者が ERP の一方で「確認中」を見て、別の機能では出荷を止める警告に出会わないまま先へ進めることだった。

「デモはここで止めましょう」と Alice が言った。

誰も反対しなかった。実際の顧客データには触れておらず、変更は本番環境にも出ていない。まさに、そのためのテストだった。

しかし、そのあとの沈黙は、成功したデモのときほど心地よいものではなかった。

状況を説明されて、Vivien も通話に加わった。

Lucas に自分の判断を説明してもらい、それから Thomas にも同じように話してもらった。テスト内容や各変更の範囲について質問した。

「二人とも Mini PRD はある？」と聞いた。

「あります」と Lucas。

「判断のルールも？」

「あります」と Thomas。「既存のブロックの条件は変えていません。出荷準備という限られた範囲の改善です」

「僕も同じです」と Lucas。「不確かなことを隠さないよう情報を追加しました。それが重要だと書いたはずです」

Vivien は二つの文書を並べて開いた。

どちらも間違ってはいなかった。ただ、違うことを扱っていた。

一方はロット追跡で不確実性をどう示すかを記述していた。もう一方は出荷をいつ停止すべきかを記述していた。二つを結ぶ問い、つまり、あるロットに疑いはあるが明示的にブロックされてはいないとき、ソフトウェアがどう動くべきかについては、どちらも答えていなかった。

「二つの案件の間に穴があるね」と Vivien。

Thomas はうなずいた。

「技術的には、いくつかの動作を選べます。でも、どれを望むか決めないと」

「それに、この状況で品質管理責任者が実際にどう対応しているか、顧客に聞く必要があります」と Lucas。「僕には仮説はありますが、どんな顧客にも通用するルールはありません」

Alice が口を開いた。

「求めるプロダクトの動作を明確にします。ただし私一人では決めません。サポート、Lucas、Thomas、それから結果を確かめるために顧客にも参加してもらう必要があるわ」

Vivien が顔を上げた。

「じゃあ明日、会議を設定するよ」

Alice は少しためらった。

「必要な人を集める手助けはしてもらえる。でも、私たちに代わって決める責任まで引き受ける必要はないわ」

彼を締め出したいわけではなかった。それでも Vivien には、何かが自分から離れていくように感じられた。

何年も、問題を正しい手順へ戻すのは自分の役目だった。しかし今回は、当事者たちはすでに話し合うべきだと知っていた。意見の食い違いをチケットの連続に変えてもらう必要はない。

必要なのは、どちらの開発者も一人ではできない判断だった。

翌日、Alice は全員に、何を避けたいのかを尋ねた。

Lucas は、欠けている情報があるのに、品質管理責任者が完全な追跡結果を得たと思い込む事態を避けたかった。

Thomas は、曖昧なルールによって履歴データが不完全な顧客の業務を不必要に止めたくなかった。

サポートは、仕入先から警告が届いても、直ちに製品回収になるとは限らないと説明した。対象範囲を特定し、品質部門と相談し、必要な対応を決めなければならない場合がある。

「なるほど」と Alice。「私たちは、一つの状態に二つの違う意味を持たせようとしたのかもしれないわ」

Alice は提案を共有した。

あるロットを「**要確認**」と示すことはできる。しかし、ERP が顧客企業に代わって規制や品質上の判断まで行うかのように見せてはならない。一方、そのロットを使ったり出荷したりする可能性のある手順では、警告を見えるようにし、消してしまうのでなく、定められた確認の流れに従わせなければならない。

古い履歴データの不確実性も、明確に表示し続ける。必要なデータがないことを、「危険がない」こととして見せてはならない。

「出荷機能で確認できます」と Thomas。「でも古い注文も試したいです」

「顧客と用語も確認したいです」と Lucas。「『要確認』は、あの品質管理責任者にとって同じ意味とは限りません」

Alice はうなずいた。

設計が完成したわけではない。しかし、誰が何を決めるべきかは、ようやく見えた。

仮の決定と未解決の質問を、理由や検証すべきケースとともに共有文書に書き込んだ。そして判断ルールに一つの条件を追加した。現場の判断によって複数の機能が利用する情報の意味が変わる場合、コードを元に戻せるとしても、自動的に「緑」にはならない。

「先に気づくべきでしたね」と Lucas。

「そうね」と Alice。「でも、そのためのテストでしょう。次に何を見ればいいか、分かったんだから」

Thomas は二つの文書を読んだ。

「機能を横断するテストも必要です。個別の動作を確かめるテストはたくさんある。でも、動作同士がぶつかる場合のテストはずっと少ない」

少し前から聞いていた Vivien は、メモを取った。

通話が始まってから初めて、よく知っている問題に出会った。すべての依存関係を一人で把握している人がいないのだ。

しかし、解決策が生まれる過程は以前と違っていた。Alice は画面の作り方を指示していなかった。Lucas と Thomas は作業の分割を待っていなかった。サポートが顧客の現実を伝えていた。Vivien 自身の役割は、情報の欠落を見えるようにし、対話を助けることだった。

それは確かに役立った。

だが、毎週月曜日に開く進捗会議とは、ほとんど別の仕事だった。

午後、Thomas は別チームの経験の浅い開発者からメッセージを受け取った。

古い在庫処理の中で、一見同じことを調べているチェックが二つある。なぜ二度確認するのか分からない。コードエージェントは一つを削除するよう提案していた。

Thomas はファイルを開いた。

「一緒に確認するまで触らないで」と返した。

二人は画面を共有して三十分調べた。二つ目のチェックは、昔の製品変更から受け継がれた特殊なケースを扱っていた。コードには十分な説明がなかった。Thomas 自身、その変更がいつのリリースに入ったか正確には覚えていなかったが、調べる場所を知っていた。

「AI が勝手におかしな答えを作ったわけじゃないんだ」と説明した。「似たチェックが二つあるから、単純化できると提案した。ソフトウェアの歴史を知らなければ、そう考えるのも分かる」

「Thomas は AI を使ってコードを書かないんですか？」と若い開発者。

「こういう部分を、よく分からないまま書き換えさせることはしないね」

「でも三十分かかったでしょう。Lucas なら五分で見つけたかもしれません」

Thomas は少し考えた。

「そうかもしれない。Lucas も同じ答えにたどり着いたかもしれない。でも大切なのは、君がなぜそのチェックがあるのか理解することだ。次は AI にも僕にも、もっと良い質問ができる」

若い開発者は説明を文書にしてよいか聞いた。

「ぜひそうして。覚えているのが僕だけというのは、もううんざりだから」

通話を終えた Thomas 自身も、最後の言葉に驚いていた。

長年、例外についての知識が彼を欠かせない存在にしていた。AI の登場でその経験が無価値になるのではないかと、不安に思ったこともある。しかし今の会話は、別の可能性を示した。すべての答えを自分だけが持つのでなく、他人が必要な判断力を身につける助けができるのだ。

Thomas は共有文書を開き、数行を書き加えた。

月曜日の定例会議で、Hubert は実験の進み具合を Alice に尋ねた。

Alice は、両立しなかった二つの判断、それを見つけたテスト、そしてルールの修正を説明した。

「つまり、二人の開発者がそれぞれ決めた結果、整合性を失いかけたということだね？」と Hubert がまとめた。

「はい」

「安心できる話じゃない」

「本番環境へ出す前に見つけられました」と Alice。「問題は、二人が決めたこと自体ではありません。二つの決定が交わるところを、私たちのルールが扱っていなかったことです」

Hubert は問題の記録を見た。

「前なら承認会議をもう一つ開いていただろう」

「かもしれません。でも会議を増やしても、あの問題に気づいたとは限らない。全体の流れを試す人と、複数の分野にまたがる判断を誰が担うかの明確さが必要だったんです」

「今は誰が決める？」

「Lucas は自分の案件を担います。Thomas は別の案件を担い、技術的なリスクについても私たちを助けます。私はプロダクトの整合性に責任を持ちます。慎重さが必要なケースでは専門家に相談します。すべてを道具に任せるわけにはいきません」

Hubert はゆっくりとうなずいた。

「Vivien は？」

Alice は少し考えてから答えた。

「二つのチームを結びつけ、どこで問題が起きているか理解する手助けをしてくれました。でも、仕事を割り振ったりプロダクトの動作を決めたりする必要はありませんでした」

Hubert の視線は画面に止まったままだった。

「金曜日に Vivien と話したんだ」と言った。「新しい進捗管理を見せてくれた。報告はとても良い。でも、話してくれた内容の八割くらいは機械に準備できる気がする」

「Vivien 自身も分かっています」と Alice。

「それをどう考えるべきか、私には分からない」

Alice にも完成した答えはなかった。

「だから彼が不要になった、ということではありません」と、ようやく答えた。「今見たように、人と人が協力する助けができます。ただ、すべての仕事が彼を通らなければならない役職では、もうないのかもしれません」

Hubert は画面を見つめたままだった。

「彼と話す必要があるね」

Vivien は翌日の会議への招待を受け取った。

最初は組織再編の話だと思った。ここ数週間、予定表に空白が増えていた。開発者は自分で案件を動かし、以前は夜にまとめていた情報の一部はツールが自動で集めていた。人間関係の難しさに対処し、新人を迎え、複雑な問題を解く仕事はまだあった。しかし、それだけで以前ほど予定は埋まらなかった。

Malorean のリリースに欠かせない存在であることを、Vivien は誇りに思っていた。

以前ほど必要とされなくなったと知るのは、喜ばしいことなのか分からなかった。

ところが会議は、予想外の質問から始まった。

「これからも何を続けたいと思う、Vivien？」と Hubert。

「私の仕事です」と答えた。

Hubert は微笑んだ。

「その仕事が何なのかを考えたいんだ」

Vivien は、チームの状況を話すために参加していた Alice を見て、また Hubert を見た。

「きれいごとは言いたくありません。チームはもう、毎週私に進捗を聞いてもらう必要はありません。それにエージェントが準備した要約を、私が読み直すだけの仕事を続けたいとも思いません」

「でも、まだ何が足りない？」と Alice。

「どこにも記録されない対立です。困っていると言い出せない人。誰に質問すればいいか分からない新人。複数チームにまたがる判断。それから、ただ時間を取って話を聞く人が必要なこともあります」

Vivien は少し間を置いた。

「ただ、それらを全部、これからも私が担当しなければならないわけじゃありません」

Hubert は姿勢を正した。

「そうだね。君の役割が変わるなら、それぞれに誰が責任を持つか、明確にしないといけない。役職名だけなくして、誰も人を支えなくなっていた、という事態は避けたい」

Vivien は同意した。

三人は開発者の育成、チーム間の判断、経営側が引き続き見るべきことについて話した。Alice は、古いコードの理解を深めようとする同僚たちを Thomas が支えていることを説明した。

会議から決定的な答えは出なかった。

終わったあと、Vivien はしばらく予定表を眺めた。

前日より脅かされている感じは減っていた。でも、安心したわけでもない。自分の仕事は消えていなかった。ただ、いくつもの責任に分かれ、その一部は違う形で担えるようになっていた。

その晩、Vivien はいつもより早くパソコンを閉じた。

久しぶりに、遅れたリリースを取り戻す仕事はなかった。

そして、取り戻した時間を何に使うべきか、まだ分からなかった。

10 – AI によって企業の中間管理職は消えるのか

今ではチームが決定事項を直接確認し、既知のルールの中で行動し、進捗の確認の一部をエージェントに任せられる。情報を伝えてもらったり、作業の一段階ずつを段取りしてもらったりするために、責任者を通す必要は以前より少ない。

では、**そうした役割を担ってきた階層はどうなるのだろうか。**

一見、答えは明らかだ。AI が調整を引き受ければ、中間管理職は不要になる。しかしマネージャーは、指示を伝えるだけの人ではない。難しい選択をし、人を支え、チームを結びつけ、どの手順にも当てはまらない問題を引き受ける。

本当の問いは、何人の役職が消えるかというよりも、**これまで担っていた仕事と権限がどこへ移るのか**である。

意思決定を調整するために作られた階層

重要な顧客から不具合の報告が来る。サポートが責任者に伝え、その責任者がプロダクト部門に連絡する。プロダクトは Engineering へ見積もりを求める。複数の責任者が情報を持ち寄り、緊急性を比べ、必要ならさらに上の階層へ判断を求める。

この流れが必ずしも不合理というわけではない。それぞれの責任者が背景情報の一部を持ち、会社の一定範囲に責任を負っている。しかし、問題を理解して決めるために、この長い連鎖を通る以外に方法がないなら、負担は大きくなる。

歴史的に、階層組織は情報が分断された状態でも企業を大きくする助けになった。中間層は経営の目標を優先事項へ落とし込み、チームを調整し、依存関係を見て、人員や予算をめぐる対立を解決してきた。社員の評価、成長、対人関係の問題にも関わっている。

ただし、これらの仕事は性質が違う。障害の報告書を書くことと、商業的な機会を見送ると決めることは同じではない。納期を追う仕事には、対立を解いたり誰かの成長を助けたりするときと同じ判断力は必要ない。

以前 Malorean で Product Owner について書いたときも、似た区別をした。単なる連絡役としての役職の必要性が薄れても、その役職が果たすべき機能まで不要になるわけではない。肩書きではなく、実際に行っている仕事を見るべきなのだ。[1]

中間層を減らすなら、そこで行われていた仕事を誰が担うのか。

リモートワークはすでに調整のルールを変えていた

この問いを AI が初めて生み出したわけではない。離れた場所で働き、ときに複数の国や時間帯をまたぐ企業は、廊下での会話や急な会議に頼らず決める方法を学ばなければならなかった。

GitLab の「handbook-first」は、ルールと背景情報を文書で誰でも確認できるようにする考え方だ。誰でも修正を提案でき、分野の責任者が採用するかどうかを判断する。社員は、マネージャーに聞くまで情報を待つ必要がない。[2]

Alan は違うが補い合う方法を紹介している。重要な決定を GitHub のスレッドで議論する。一人が決定して実行し、別の人が案件の成功に責任を持ち、関係者は相談を受けるか情報を共有される。全員一致でなくても議論を公開できる。[3]

Basecamp を生んだ 37signals では、調整のかなりの部分を文書による対話、振り返り、プロジェクトとともに保存される情報で行い、すぐ消える会話に散らばらせない。[4]

これはマネジメントがない企業三社の例ではない。**情報を独占する人や、決定事項を何度も伝え直す人への依存を減らす三つの方法**だ。

この先例は AI エージェントにとって重要だ。ルールや決定、責任者が明示されていれば、仕組みはそれを検索して利用できる。マネージャーの頭にしかない決定は、同僚にもエージェントにも利用できない。

とはいえ、リモートワークを理想化してはならない。Microsoft の六万千人を超える社員を対象とした研究では、在宅勤務への移行後、協働のネットワークがより分断されたことが観察された。たくさん書けば情報共有やチーム間の協力が自然に生まれるわけではない。[5]

リモートワークから学ぶべき本当のことは、オフィスをなくすことではない。常に誰かが仲介しなくても利用できる情報と決定を中心に、仕事の一部を組織できることだ。AI はこの土台を利用できる。しかし、それを AI だけで作れるわけではない。

AI は、この仕事の一部を引き受け始めている

文書はルールを残す。ダッシュボードは遅れを示す。エージェントはさらに進み、複数の情報を照合し、進捗をまとめ、矛盾を見つけ、対応を提案し、許可された行動を実行できる。

これらはまさに、多くのマネージャーが時間を使ってきた仕事だ。情報の集約、報告、進捗管理、判断の準備である。そのコストは下がり得る。ただし、仕事が完全になくなると決まったわけではない。

HEC Paris による、45 件のインタビューなどをもとにした探索的な分析では、構造化されデータが豊富な管理業務は、信頼、人間関係、判断力に依存する業務より自動化しやすいと示唆された。しかし、実際に削減された管理時間や不要になった役職の数を測定したわけではない。[6]

Procter & Gamble で行われた実験でも、製品イノベーションの課題で、一人が AI と働くことで、AI を使わない二人組に匹敵する結果を出せる場合があった。AI は異なる専門分野の視点を組み合わせる助けにもなった。だが一度の協働を助けることと、長期にわたって責任者を置き換えることは違う。[7]

別の実験では、66 社の 7,137 人が対象となり、AI アシスタントを定期的に使う人は、研究後半に週約二時間のメール作業を節約した。一方、六か月間で仕事全体の構成が有意に変わったとは観察されなかった。これは事務作業用アシスタントの研究であり、プロセス全体を自律実行するエージェントの研究ではない。[8]

この隔たりは決定的だ。**一人ひとりの生産性が上がったからといって、それだけで組織が変わるわけではない。**

中間層の必要性を下げるには、情報が実際にチームへ届き、意思決定の権限が変わり、想定外の事態を判断する場も必要だ。道具の問題ではなく、組織の問題なのである。

三つの企業、まったく異なる三つの方向

どの企業も同じように変わっているわけではない。三つの事例から、すでに進んでいる組織再編、AI を中心に構想された目標、そして急速な変化の難しさを区別できる。

Bayer：実際に減らした階層

Bayer は*Dynamic Shared Ownership*（DSO）というモデルで、判断の一部を部門横断のチームへ分配した。2025 年の株主向け書簡で経営者は、最大六つの組織階層をなくし、管理職のポストを約三分の二減らしたと述べた。意思決定が速くなり、開発期間も短縮されたという。[9]

影響の大きさは注目に値する。ただし成果は Bayer 自身による報告だ。この変革では組織図の肩書きだけでなく、日々の責任を変える必要があった。現場チームがより多く決め、責任者は育成へ力を使い、調整業務の一部が組み直された。

重要な点がある。**DSO は生成 AI エージェントが広がる前から始まっていた。** Bayer は大企業が階層を大きく減らせることを示すが、AI がその原因だった証拠にはならない。

Block：階層を通さずに背景情報を届ける

Block は、さらに大きな役割を AI に与えようとしている。Jack Dorsey と投資家 Roelof Botha が共同で書いた文章では、共有の情報システムを基盤とする組織が描かれている。

社員が責任者から説明を待たずに状況を理解し、行動できるようにするのが目標だ。執筆者は、個々の貢献者、横断的な成果に責任を持つ DRI、そして自分も成果を出しながら人を育てる*player-coaches*を区別している。[10]

情報の伝達役としてのマネージャーの重要性は下がるかもしれない。それでも、人の育成、結果への責任、経営上の決定は残る。

これは企業と投資家が掲げる構想であり、そのモデルが長期にわたって成功したと独立に証明された結果ではない。

Meta：生産量が確認能力を超えたとき

対照的なのが、2026 年の Meta の組織再編計画について Reuters が報じた調査だ。より小さなチーム、AI の活用、一部の階層の削減が計画または実施されていた。最終的には第二波の再編が中止された。

Reuters が引用した社内文書によると、社内ソフトウェアとインフラへの変更は、利用者向けの新機能よりはるかに速く増えていた。同時に、技術的な障害と、その解決に必要な時間も増えていた。Meta は報道の一部に反論し、説明を補った。[11]

この数字だけでは、障害の原因を AI や管理職の削減と断定できない。同時に複数の変化が起きていたからだ。しかし、重要な問いを投げかける。作る量が増えても、確認し、安定させ、受け止めることができなければ、何の意味があるのだろう。

Bayer は階層を減らせることを示し、Block は AI を前提とする組織の姿を描き、Meta は生産量が調整能力を超えた変化の危険を示す。どれも、それだけでマネジメントの未来を予告するものではない。

管理職の役職がなくなっても、仕事はなくならない

例えば顧客がソフトウェアの不具合を報告したとする。既知の問題で対処のルールが明確なら、チームは長い承認手順を待たずに動ける。情報が信頼できれば、調べ、修正し、関係者へ伝えられる。

しかし二つの診断が矛盾したらどうだろう。エージェントはある対応を勧め、開発者は別の方法を勧める。修正が隣接するサービスを壊すかもしれない。

そのときは誰かが意見の違いを明らかにし、必要な専門家を集め、どのリスクを取るか決めなければならない。自律性が調整をなくしたのではない。調整の方法を変えたのだ。

Martin Krzywdzinski らの実験は興味深い手がかりになる。模擬生産ラインで 87 組の二人組が、ときどき誤った助言をする AI の助けを受けながら不具合に対処した。自律的に組織された組では、正式なリーダーを置いた組より多く対話し、問題をよりよく解決した。研究ではコミュニケーションの重要性が強調されている。[12]

研究室の二人組を大企業と同一視することはできない。しかし一つの区別を思い出させてくれる。**正式なリーダーなしで働くチームはあり得る。それでも調整するという仕事は必要だ。**

調整は同僚、専門家、決定権を委ねられた人、階層上の責任者のいずれも担い得る。適切な形は技能、リスク、対立の種類によって変わる。

自動生成された報告書だけではできないことも残る。新しい人を迎え、育て、緊張関係を解き、難しい決定を支え、努力を認めることだ。こうした仕事をすべてのマネージャーがうまくできているとは限らず、別の形で分担することもできる。しかし誰かが、その使命と時間を持たなければならない。

Gallup の調査では、AI を使っていても雇用への不安は消えず、職場で支えられているという感覚は不安の少なさと関連していた。伝統的なマネージャーの必要性を証明するものではないが、組織が人間の集まりでもあることを思い出させる。[13]

そして、将来難しい決定を担う人を、誰が育てるのか。

私は Malorean で、若手開発者へのリスクも取り上げた。AI がコードをすぐ書き、誤りをすぐ直してしまうと、デバッグし、失敗を理解し、判断力を培う機会が失われるかもしれない。研究すべき懸念であり、技能が全体として失われているという証拠ではない。[14]

未来の責任者にも同じ問題がある。判断に同席し、フィードバックを受け、チームが対立を乗り越える様子を見ることで学ぶこともある。階層を減らす企業は、経験を伝える別の道筋も用意しなければならない。

結局、一つの管理業務にも複数の行き先がある。自動化、チームによる分担、専門家への移管、経営層への集中だ。**仕事は消えることも、担い手が変わることも、誰かの仕事に追加されることもある。**

だから組織の階層を減らした効果を、消したポストの数だけで測ることはできない。省いた進捗管理と、新たに生じるエージェントの監督、障害対応、育成、支援の負担を比べる必要がある。この再分配の全体的な収支について、まだ十分に確かな数字はない。

階層が減れば、本当に自律的になるのか

組織図からは見えにくい最後の問いがある。**仲介する人がいなくなったとき、実際に誰が決めるのか。**

情報を得たチームが、多くの問題を上司に上げずに解ける姿を想像できる。しかし逆もあり得る。現場の責任者は減っても、システムを通じて経営層がより多くの判断を直接管理する姿だ。

この緊張関係は、生成 AI 以前からあった。Nicholas Bloom らの製造企業についての研究では、社員に情報を与える道具が、現場での自律性の増加と関連していた。一方、上司との連絡を容易にする道具は、中央集権化につながる場合があった。トップへ指示を求めやすくなるからだ。[15]

現代の AI エージェントについて何かを証明する研究ではない。それでも一つの仕組みを明らかにする。**情報を広く届けることと、権限を広く分けることは違う。**

アルゴリズムによるマネジメントは、この危険をさらに具体的にする。システムが仕事を割り振り、優先順位を提案し、成果を測り、警告を出すことができる。研究では、人間のマネージャーが残っていても、指揮や評価の機能の一部をこうした道具が担う場合があると示されている。[16]

システムは組織図に上司としては現れない。しかし、誰がどの仕事を受け取り、何を基準に評価され、どんな条件で行動を許されるかを決めるなら、権限の一部を形づくっている。

見るべきなのは、画面の背後にある決定だ。エージェントの目標を誰が設定するか。ルールを誰が選び、変えられるか。誰が異議を唱えられるか。アルゴリズムが間違えたとき、その行動を本当に取り消せるのは誰なのか。

チームは大きな実行能力を持ちながら、優先順位を変える力は持たないかもしれない。一方、マネージャーがいても、方法と現場の判断を本当に自分たちで決められるチームもある。

私は Malorean で、選択や受け入れる損失、最終決定者を明らかにせず、見かけ上の合意だけを作る「曖昧な決定」を取り上げた。権限の所在が同じように不明なら、階層を一つ消しても問題は変わらない。[17]

つまり、**見える階層は減ったのに、道具の中では管理が中央へ集中する**という逆説が起き得る。

必然ではない。同じシステムが現場での判断を助け、権限のルールを透明にすることもできる。違いを決めるのは、組織図から消した箱の数ではなく、企業が実際に何を委ねるかだ。

本当に消えるかもしれないもの

AI によって企業の中間管理層は消えるのだろうか。

必要性が下がる階層はあるだろう。しかし、それは自動的な変化でも、マネジメントの終わりでもない。

情報を伝え、進捗を集計し、繰り返しの承認を段取りすることが中心の仕事は、比重を下げるかもしれない。一方、難しい判断、対立の解消、経験の継承、結果への責任といった仕事は、別の形で担わなければならない。

Bayer は AI を原因とせずに大きく階層を減らせることを示した。Block は調整の一部を共有システムへ移す姿を描く。Meta は信頼性や人間の負担を測らずに実行を速めると、組織が不安定になる危険を示す。AI エージェントによってマネジメントが全般的に消えるという証拠はまだない。

それでも変化は深いものになり得る。情報や承認を得るために、必ず通らなければならない人としての責任者は、以前より必要なくなるかもしれない。その分、一貫性、例外的な問題、人の成長に注意を向けられる場合がある。別の企業では、それらの仕事の一部をチームが直接担い、ソフトウェアに組み込んだり、少数の決定者へ集中させたりするだろう。

不要になるのはマネジメントそのものとは限らない。調整するためには必ず階層を通らなければならない、という仕組みが問い直され始めているのだ。

開発者にとっても、これは現実的な変化だ。仲介する人が減れば、単に作るための道具が増えるだけではない。ニーズや妥協、そして責任を負うべき結果に、より直接向き合うことになる。

これこそ、この本で見てきた動きだ。行動できる力が増えるほど、理解し、選び、結果を引き受ける必要も高まる。

それで判断が良くなるとも、組織が人間的になるとも限らない。ただ、違う形で組織を設計する必要が生まれる。背景情報を共有し、権限の所在を見えるようにし、経験を継承し、成果に責任を持つことだ。

問題は、AI によって何を作れるようになったかだけではない。その新しい能力を使って、どんな企業を作りたいのかということだ。

最後のエピソードでは Malorean Corp.へ戻り、経営者と社員が組織図の向こう側で、自分たちの選択をどう生きているか見ていこう。

参考文献

[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.en.html
- 単なる伝達役としての職位と、本来果たすべき機能を区別する著者の論考。

[2] Handbook First – GitLab

- URL： https://handbook.gitlab.com/handbook/company/culture/all-remote/handbook-first/
- 文書を共有の基準とし、公開された形で改訂を議論しながら、人間の責任者も残す運用。

[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

- 非同期で決定を議論する方法、決定者、責任者、相談・連絡する相手について説明する。

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

- URL：https://37signals.com/how-we-communicate
- 書面による非同期のやり取りを中心に、長く利用できる情報を残す社内コミュニケーション。

[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/
- リモートワーク移行後の協働ネットワークを研究し、分断が強まる危険を示した調査。

[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
- インタビュー等を用いた探索的分析。実際に失われた管理職の数を因果的に測ったものではない。

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

- URL：https://www.nber.org/papers/w33641
- P&G のイノベーション課題で AI が専門分野の視点を組み合わせる助けになるかを検証した実験。

[8] Shifting Work Patterns with Generative AI – NBER

- URL：https://www.nber.org/papers/w33795
- 66 社の無作為化研究。個人の時間節約はあっても、六か月間で仕事全体の構成には有意な変化がなかった。

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

- URL： https://reports.bayer.com/annual-report-2025/en/to-our-stockholders/chairmans-letter.html
- AI エージェントの普及以前からの DSO 改革で、Bayer が報告する管理階層と管理職の削減。

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

- URL：https://sequoiacap.com/article/from-hierarchy-to-intelligence
- 共有情報システムと DRI、player-coaches によって調整する組織の構想。

[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/
- TimesLIVE に転載された Reuters の調査報道。Meta の組織・技術変革に伴う緊張や問題を扱う。

[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
- AI が関与する模擬設備の障害対応を通じて、二人組の組織の形と対話の役割を比較する実験。

[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
- AI についての不安、職場での尊重と支援の感覚に関する調査。

[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.en.html
- AI に修正と実装を任せることで、若手開発者が学ぶ機会を失うかもしれないという著者の考察。

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

- URL：https://www.nber.org/papers/w14975
- 社員への情報提供と、上司への連絡を容易にする技術が、現場の権限に与える違った影響を扱う。

[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
- アルゴリズムによる指揮・評価の機能と、人間が介入できる範囲についての研究。

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

- URL： https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.en.html
- 一見合意が成立したようで、実際の選択と責任者を曖昧にする決定についての著者の論考。

11 – Malorean Corp. – 取り戻した時間

休暇管理機能のデモから二か月後。月曜日の会議で、Hubert Félix Duca は開発の進捗を尋ねなかった。

「さて、先週は何を学べたかな？」

Alice は画面を共有した。

そこには、最近の顧客の声の要約、進行中の実験の結果、まだ決めていないことがいくつか並んでいた。夜の間にエージェントが情報をまとめていた。Alice は朝、それを読み、誤解を生みそうな結びつけをいくつか削り、エージェントだけでは思いつかない問いを二つ加えていた。

「ロット追跡では、データが完全な場合、試験導入先の顧客は関係する製造記録をずっと速く見つけられるようになっています」と話し始めた。「ただ、過去の記録には限界を明示しなければならないものも残っています。品質管理責任者と新しい表示を試す予定です」

Hubert はうなずいた。

「ほかの顧客では？」

「結論を出すには早すぎます。だからまだ全顧客には展開していません」

続いて Bob が話した。別の製品案件を担当し、販売前の支援チームやサポートと何度も話したばかりだった。

「見た目は似ている要望が三つあります。エージェントは一つの機能案としてまとめましたが、背後に少なくとも二つの違う問題があります。投資を決める前に、対象を明確にしたいんです」

「開発者は何人必要？」と Hubert。

Bob は笑った。

「今のところゼロです」

Hubert も笑い返した。

Bob は通知を見て付け加えた。

「そういえば、昨日からサポートが、繰り返し届く要望を新しい方法で分類するテストをしています。チームの一人がエージェントと一緒に、匿名化したコピーを使って、正しい担当者へ情報が届くまでの時間を短くできるか試しています」

「誰が始めた？」と Hubert。

「サポートです。Alice が、社内だけで行い元に戻せる範囲で、人間が結果を確認することを確かめています」

「君は知っていた？」

「アイデアは知っていました。テストの細かい内容までは」

Hubert は、なぜ事前に知らせなかったのかと反射的に聞きかけたが、思いとどまった。小さく、元に戻せる取り組みには自分の承認を不要にすると、合意したばかりだった。

「結果が出たら教えてくれ。続けるなら、誰が維持するのかも決めたい」

テストを許可する必要はなかった。ただし、広く導入するなら、そのための資源に責任を持つのは彼だった。

少し前なら、プロダクトの会議でこの答えは奇妙に聞こえただろう。主な関心は、開発者が何をどんな順番で作るかだった。今では Alice と Bob は、解くべき問題を選び、取り組みの整合性を確認し、既存のルールの範囲を超える決定に関わることへ、より多くの時間を使っていた。

肩書きは変わっていない。人事上は、今でも Product Manager だった。

しかし二か月前とは、日々の仕事が少し違っていた。

最も目立つ変化は、もはや話題にすらならないことだったのかもしれない。

開発者が顧客と話す。別の開発者が、自分の担当機能の改善に Mini PRD を提案する。営業担当者が専用の環境でデモの別案を作る。サポートの人が、プロダクトのルールの矛盾を直接指摘する。

もう、職種の境界について大きな議論にはならなかった。

Malorean は、誰もが何でもできると発表したわけではない。ただ、ある行動は必ず従来の担当職種を通らなければならない、という前提をやめた。

五人ずつの二つの開発チームは今も存在していた。その多くが、いくつかの案件を最初から最後まで担うようになった。問題を理解し、課題の整理を提案し、関係者と話し、開発し、テストし、結果を追う。

この働き方になじむ速さは人によって違った。二人の開発者は、課題を整理するときに今でも丁寧な支援を望んでいた。技術的に複雑な問題には非常に優れていても、顧客との会議を増やしたいと思わない開発者もいた。誰も彼らを無理に Lucas のように変えようとはしなかった。

チケットも残っていた。

作業の調整、具体的なタスクの記録、修正状況の追跡に使っていた。しかし、案件の存在理由、過去のすべての決定、エージェントに必要な情報を、無理に一つのチケットへ詰め込むことはしなくなった。

代わりに、各案件の責任者が、共通の文書ツール内で背景情報を管理するようになった。目標、決定、資料、リスク、未解決の問い、そして既存のチケット、見積もり、サポートの声、面談メモ、テスト、時には発掘した昔の議論へのリンクが置かれていた。

すべてをコピーする必要はなかった。大切なのは、なぜそう決めたかを理解でき、どこで情報を確かめられ、誰がそれを変えられるかが分かることだった。

Alice が作った最初の文書が、手本になった。

Bob が同じ方法を取り入れ、開発者たちが続き、サポート、営業、導入支援の人たちも使い始めた。多くの人が Alice と同じように気づいた。ChatGPT に一時間話して説明すれば、自力では文書にまとめるのに何日もかかりそうな選択を、形にできることがある。

頭の中にあったことが、魔法のように全部取り出されたわけではない。それでも、理由、例外、言葉にされてこなかった知識の一部は、共有できるようになり始めていた。

リモートワークは、この動きを自然なものにしていた。新型コロナの流行以降、Malorean はフランス各地に離れている同僚が廊下での会話を待たずに仕事をできるよう、すでに記録を増やしていた。AI は知識を取り出し整理する作業を速めた。共有の必要性を生み出したわけではなかった。

Alice は*プロダクトの意思決定 — 共通ルール*というページを開いた。

「先週また言葉の意味が食い違ったの」と Bob に説明した。「あるチームの文書では『承認済み』が『技術レビューが終わった』という意味だった。でもサポートでは『顧客と動作を確認した』という意味で使っていたの」

Bob はため息をついた。

「昔から全員が同じ意味では使っていなかった言葉だね」

「そうなの。それで、ロットの問題のあとに決めた内容へリンクを追加したわ」

数週間前、Lucas と Thomas は、ロット追跡と出荷の間で両立しない二つの動作を作りかけた。変更はテスト中に止められた。それ以来、チームは一つの機能で当たり前に見える言葉が、別の機能で違う意味になることを警戒していた。

共有の背景情報は役立った。しかし、新しい問題も生んだ。

古くなったページもあった。仮の決定が、正式なルールであるかのようにコピーされたこともあった。エージェントが昔の営業上の仮説を、会社の確定した約束のように取り出してしまったこともある。

そこで規律が必要になった。各文書に担当者、出典、確認済みの決定、まだ不確かなことを記載する。重要な変更は理由を残す。機密情報をどんなツールからでも区別なく利用できるようにはしない。

「ページを増やさないで済ませることもできたかもね」と Bob。

「そうかもしれない。でも前から矛盾はあったのよ。ただ、頭の中にあって誰にも見えなかっただけ」

Alice は Bob の最新の変更を読み直した。

「原則はこれでいいわ。でもドイツの顧客についての書き方はまだ承認できない。十分なことが分かっていないから」

Bob はその行を直した。

Garant の仕事は、こういうことが多かった。あらゆる画面を承認するのではない。仮説が知らないうちにルールへ変わらないようにすることだった。

別の開発チームでは、Thomas が最近入社した開発者との会話を終えたところだった。

ERP の古いコードについて一時間説明していた。若い開発者はアシスタントを使って昔の給与計算の仕組みを理解し、一般的なルールと矛盾するように見える例外について尋ねた。

Thomas は、その例外が設けられた理由を古いリリースの記録から探し出した。

「要約だけ読むと、コードをきれいにする必要があると思うかもしれない」と言った。「でも消そうとしているものが何か、まず理解しなければならない」

「それもチームのコンテクストに書いてもらえますか？」

「今やっているところだよ」

Thomas は新しいメモを共有した。元の資料、説明、二つのテストケースが加えられていた。

Lucas のように、コードの変更をエージェントへ任せることはまだしていなかった。AI への懐疑心が消えたわけではない。しかし、新しい文書共有の仕組みは積極的に使い、同僚に経験を伝える仕事にも、以前より関わるようになっていた。

新しい開発者の受け入れを支援していた Vivien は、しばらくして Thomas にビデオ通話をかけた。

「どう？」

「いいよ。好奇心がある。何かを消す前に質問してくる」

「それは安心だね」

「それに、いくつかのチェックが存在する理由を、ようやく書く時間が取れた」

Vivien は笑った。

「じゃあ AI について考えを変えた？」

「いや。自分だけが説明を知っていればいいという考えを変えたんだ」

少し間を置いた。

「ただ、これには時間が必要だよ。人の面倒を見ろと言いながら、毎日を開発作業でいっぱいにされたら無理だ」

「Hubert には話した。計画の中に育成の時間を確保してある。ほかの仕事がすべて終わってからやる活動にはしない」

Vivien にはよく分かっていた。何年もの間、チームが「少し時間ができたら」新人を育てると言うのを見てきた。その時間は、結局やってこなかった。

二人は新しい開発者が必要としていることをもう少し話した。それから Vivien は通話を切った。

リリース管理表は一度も開かなかった。

昼どき、Hubert は Vivien を通話に呼んだ。

「一つ聞きたい。今、去年と同じ仕事に、どれくらい時間を使っている？」

Vivien は考えた。

「昔の役職をフルタイムで続ける理由になるほどではありません。それを聞きたいのであれば」

Hubert はすぐには答えなかった。

Vivien は長年、二つの技術チームの仕事の通り道だった。リリース、依存関係、一部の緊急事態、開発者と業務部門の責任者との対話を担っていた。組織が自律的でなかった頃は、毎日がその仕事で埋まった。

今は違う。

各案件には担当者が明確にいた。開発者が現場の判断の多くを下した。Vivien が毎回伝え直さなくても、情報は流れた。ツールは進捗をまとめ、依存関係を知らせてくれた。もちろん、その説明を確認する必要はあったが。

人間関係の問題、対立、育成は消えていなかった。Vivien は今も手伝っていた。しかし、日々のリリースを監督する専用の役職に、そのすべてを集中させる必要はなくなっていた。

「無駄な会議が減るのはいいことです」と Vivien は続けた。「ただ、長年チームの管理を学んできたのに、エージェントが表を正しく埋めたか確認するだけの人にはなりたくない」

「そんなことは頼まないよ」と Hubert。

二人は移行を準備していた。重要な技術的判断は、担当する Garant が行う。育成を担う人には明確な時間を確保する。Hubert はキャリアに関する選択や、文書だけでは解決できない人間関係の問題への責任を引き続き持つ。Vivien は最後に残った調整業務の引き継ぎをする。

完璧ではなかった。困っている社員への支援を互いに相手が担当していると思っていたケースがあった。誰が判断するか決まっていなかったため、一つのチームが長く待ったこともある。

そのたびに、責任分担と、それを見えるようにする方法を直してきた。

「一つだけしたくないことがある」と Hubert。「君の役職をなくして、これまでの仕事には価値がなかったかのように言うことだ」

「私も、それは嫌です」

「だから、君の経験を今どこに生かせるか考えないと」

Vivien はカップを見た。

「一つ、考えがあります」

数日前、妻がまたドイツの話をした。

彼女はドイツ人だった。二人はしばらく前から、ドイツへ移り住む可能性について話していた。急いではいない。フランスでの暮らし、習慣、友人がいた。しかし、その話題は何度も戻ってきて、次第に具体的になっていた。

Vivien はずっと先延ばしにしていた。

Malorean での仕事がとても好きだった。チームのことも、その歴史も、ソフトウェアも、顧客も知っていた。会社を離れるのは、自分が築くのを手伝ったものを置き去りにするように感じられた。

それでも、新しい挑戦は好きだった。今のキャリアの段階だからこそ、別の挑戦をしてみたいと思うことがあった。

前の土曜日、二人は互いに何も言わないまま、ドイツの住居の広告を眺めていた。やがて妻がパソコンを閉じて言った。

「ねえ、あと十年も同じ話を続ける必要はないんじゃない？」

Vivien は笑った。

だが月曜日になっても、頭から離れなかった。

「ドイツ？」と Hubert が聞き返した。

Vivien はうなずいた。

「戦略会議では、何年も話してきたことです。ドイツにも少し顧客はいます。でも、ほとんどはフランスの顧客グループの子会社です。ドイツの企業に ERP を売ることを、私たちはまだ本当には学んでいません」

「確かに」

「みんな、製品の準備ができたら進出しようと言います。でも、現地の企業と本気で話さなければ、何が足りないのか分からないでしょう」

Hubert は注意深く聞いていた。

「市場開拓を担当したいということ？」

「はい。でも一晩でドイツ市場の専門家になったふりはしません。うちの製品、制約、チーム、顧客への導入の仕方は知っています。複雑な問題を中心に人を集めることもできます。残りは学ばないといけません」

「営業の経験は？」

「Julie や Philippe のようにはありません。二人に仕事のやり方を教えるつもりもないです。一緒に働き、導入支援チームや業務の専門家とも協力して、ドイツのニーズを理解したいんです。そのうえで応えられるかどうかを見る」

Vivien は少しためらった。

「妻はドイツ人です。前から向こうに住む話をしていました。Malorean を辞めずに、本当にやってみたいことに取り組める機会になるかもしれません」

Hubert は少し黙った。

「単なる翻訳プロジェクトじゃないと分かっているね？」

「もちろんです。ERP は国ごとに違うルールや業務手順に関わります。画面を訳すだけでは足りません。何を約束できるか、何を開発・適応させるべきか、何を断るべきかを知る必要があります」

「間違うかもしれない」

「きっといくつかは間違えます。一年かけて誰も欲しがらない機能に投資したあとで分かるより、顧客と話して見つけたいです」

Hubert は案件の一覧を見た。

長い間、ドイツは「いつか」の目標だった。まず遅れた案件を終え、開発者を増やし、ERP を適応させる時間を確保し、デモを準備し、チームを教育する必要があった。

その問題がすべて消えたわけではない。しかし、一部は規模が変わっていた。

デモの準備にかかる手間は減った。開発者に必ず頼まなくても、製品の別案を試せた。サポートと販売前の支援チームは、文書を作りやすくな

り、知識も共有しやすくなっていた。開発者にも、限られた案を探るための余力が以前よりあった。

仕事がなくなったわけではない。

フランスの顧客からの要望を処理する以外に、力の一部を使うことを選べるようになったということだ。

「まず最初の一歩を提案してほしい」と Hubert。「市場へ進出すると約束する計画ではない。面談、答えを知りたい問い、必要な技能、続けるかどうかを判断する方法を整理して」

Vivien は笑った。

「ドイツ向けの Mini PRD ですか？」

「からかわないでくれ。技術的な案件をこの方法で整理できるなら、市場の仮説についても試す価値はあるだろう」

「分かりました。でも、私たちが間違っていたと発見する権利は残してください」

「もちろん、それを期待しているよ」

翌週、Hubert は Alice、Bob、Vivien、Julie、Philippe を集めた。

何を開発しなければならないかを聞く前に、提案を紹介した。

「二か月前にはなかった能力が、今の私たちにはある。その一部を使って、ドイツ市場に本格的に入るとは何を意味するのか理解したい」

最初に反応したのは Julie だった。

「ドイツ語のデモならもう準備できます。見込み客へも、かなり速く連絡できると思います。AI があれば数時間で資料を現地向けにできます」

「資料はそうかもしれない」と Philippe。「でもドイツの経営者が ERP を買うとき、プレゼンがドイツ語だからという理由だけでは選ばないよ」

Julie は腕を組んだ。

「そんなことは言っていないわ」

「分かっている。どうやって仕入先を選ぶか、誰に相談するか、どんな信頼を求めるか理解しなければならないということだ。僕たちはまだ市場を十分に知らない」

ツールの議論に変わる前に、Vivien が入った。

「まさに、それを知りたいんです。Julie には案やデモをすばやく準備する力が必要です。Philippe には大口顧客の経験と、きれいな画面だけでは解決できない交渉の知識が必要です。導入支援の人たちは、顧客が後から知って困る問題を理解しています」

Alice が話した。

「私たちは製品の限界について明確にする必要があります。速く探索できても、確認していない法規制対応の機能を約束するわけにはいきません」

Bob もうなずいた。

「最初にどんな企業を理解するか決めないといけません。『ドイツ』は顧客の種類ではないからです。全員に応えようとすると、フランスですでに学んだ失敗を繰り返します」

Hubert は画面の顔を見た。

Vivien に方法を提案するよう頼んだ。今すぐ常設チームを組めとか、すべての開発能力を確保しろとか、商業展開の日付を約束しろとは言っていない。

それでも何人もの人が、すでに自分にできる貢献を考え始めていた。

Malorean では、それが習慣になっていた。

「よし」と Hubert。「決定を伝える。最初の探索段階に予算を付ける。Vivien が担当する。ドイツ企業のどの問題は今すぐ解けるか、どれには適応が必要か、どれは引き受けたくないかを明らかにしてもらう。そのあと、さらに投資するか決めよう」

付け加えた。

「それから、道具で時間が節約できたという理由だけで誰かをこの企画に移すことはしない。今の顧客を見捨てずに本当に使える時間がどれだけあるか、確認する」

Vivien は最後の点を書き留めた。何年も計画を管理してきたので、表の上で空いた時間が現実には使えないこともあると知っていた。

それから数日、Vivien は販売前の支援チームと導入チームの人たちに話を聞いた。

再利用できる設定を使ってデモを準備できるようになり、顧客との対話も簡単に要約できるようになったため、彼らの仕事の一部は変わっていた。二人が、同じ資料を何度も作り直す時間が減ったので、ドイツで最初の面談に参加したいと言った。

「導入についての質問を準備するのを手伝えます」と一人。「遠隔でいくつかの面談にも参加できます」

「フランスの顧客は？」と Vivien。

「全部放り出すことはできません。でも予定に組み込めば、時間を確保できます」

一方、難しい導入案件をいくつも抱えている別の人は、まったく余力がないと答えた。

Vivien は無理に頼まなかった。

暫定的な分担を決めた。ドイツ企業の期待を理解する時間を少し予約し、役立つ場合にはすでに顧客となっている現地子会社と話し、実際の負担が分かるまで誰にも常時参加を約束させない。

節約できた時間が、自然に会社の成長へ変わるわけではない。その時間を何に使うか選び、空いた一時間を昨日までの要望で即座に埋めないことを受け入れなければならない。

ある晩、Alice は Lucas からメッセージを受け取った。

ドイツの企画を見ました。業務手順の違いを試すのに役立ちそうな設定機構がすでにあります。標準の製品に触らずに、探索用の環境を準備できます。興味ありますか？

Alice は読み返した。

二か月前なら、似た提案は、開発者は依頼されるまで案件を始めてはいけないのかという議論を呼んだだろう。今の最初の問いはもっと単純だった。戦略に役立つか、どんな条件なら進められるか、だ。

Alice はドイツ探索の第一段階の共有文書を開いた。Vivien がすでに仮説、責任者、制約、未解決の問いを書いていた。

返信した。

いいわ。*Vivien* と範囲を決めて、業務上の仮説を検証なしで標準機能にしないこと。何を作れるかではなく、何を学びたいかから始めて。

Lucas からすぐ返事が来た。

了解です。*Mini PRD* を用意します。

Alice は笑った。

二日で作った休暇管理機能を Lucas が見せた金曜日を思い出した。あのときは誰もが、どうやってこれほど速く大量に作れたのか理解しようとしていた。

今では、彼がデモをすばやく作れることに驚く人はいない。

重要なのは、なぜそのデモが必要なのかを知ることだった。

Vivien は、ドイツ企業との最初の面談の日付を妻と共有して、一日を終えた。

まだ契約は一つも取っていない。Malorean が現地オフィスを開いたわけでもなく、ERP をドイツ市場のすべての要求に対応させたわけでもなかった。現地の慣行を知り、法律と規制の制約を確かめ、自分たちを知らない企業に信頼してもらう方法を学ぶ必要があった。

確かなことより、分からないことのほうがずっと多かった。

「怖い？」と妻が聞いた。

「少し」

「チームをまとめていたころは、自信たっぷりだったのに」

Vivien は笑った。

「何年もかけて、その仕事を学んだからね」

「今は？」

彼は最初の訪問のために書き始めた文書を見た。

質問。仮説。会うべき人の名前。まだ知らないことの最初の一覧。

「今は」と答えた。「また新しく学ぶことができたんだ」

妻は近づいて画面を見た。

「それで、私たちのことは？」

「引っ越しについて、本気で話せるようになるね」

妻は彼の肩に手を置いた。

Vivien は昔のリリース管理表、月曜日の会議、案件を一つに結びつけてきた仕事を思い出した。その仕事が好きだった。これまでの年月を後悔していない。

チームがもう同じようには自分を必要としていないと受け入れたことも、後悔していなかった。

画面には、最初のドイツ企業との面談が確定したと表示されていた。

Malorean にとって、それはまだ市場の獲得でも成功でもなかった。

新たな出発だった。

12 – おわりに

職種の境界は薄れ、責任はより明確になる

職種によって、一人ひとりにできることの範囲が決まる時代ではなくなりつつある。開発者が顧客の課題を調べ、PM が自ら作り、営業担当者が試作品を作れる。AI はかつて他の専門分野に限られていた仕事への道を開く。しかし、それで専門性が互いに置き換えられるようになったわけではない。

私たちはもはや、自分の手で実行したことだけに責任を負うのではない。複数の機能をまたぐ案件を主体的に担う一方で、各分野の「保証責任者（Garant）」が、その専門領域の一貫性を守る。行動できる人の範囲は広がっても、責任まで曖昧にしてはならない。

危険なのは二つの極端な考え方だ。「それは私の仕事ではない」と言い続けること。そして「自分でできるのだから、一人で決めてもよい」と思い込むこと。職種ごとの縄張りを守るだけでも、境界を完全になくすだけでも、仕事はうまく組織できない。

自律性を支えるのは、共有された背景情報

自由が増えて意味を持つのは、一人ひとりが適切に判断できる材料を持っている場合だけだ。背景情報が共有されていなければ、ある場所では正しく思える判断が、より大きなニーズや方針、制約に反することがある。

だから、記録を残すとは議事録を積み上げることではない。解くべき問題、目的、仮説、決定の理由、制約、どの段階で上位判断を求めるかというルールを、誰でも見つけられるようにすることだ。こうしたことを明確にすれば、一歩進むたびに情報を知っている人を探さずに行動できる。文脈や前提情報は、人間にも AI エージェントにも、判断を支える道具になる。

保証責任者の働き方もそこで変わる。すべての取り組みを点検するのではない。判断のためのルールを分かりやすくし、その整合性を守り、既存の範囲を超える決定が必要なときに関わる。適切な歯止めがなければ

自由は混乱につながる。何でも承認を求めなければならないのなら、自由は口約束にすぎない。

マネジメントの仕事も変わる。進捗の確認や調整の一部は、ほかの方法で担えるようになる。それでも、人を支え、経験を伝え、対立を解き、難しい判断を引き受ける必要は残る。それを形づくっていた従来の作業がなくなっても、責任そのものが消えるわけではない。

本当に大切なことを、もっとできるようにする

この変化を、作れた機能の数や節約した時間で測りたくなるだろう。確かにAIによって、生み出せる量は増える。しかし、空いた時間をすぐ別の仕事で埋めるだけなら、これまでの働き方を目的を問わずに速くすることになる。

それよりも、顧客のニーズを深く理解し、仮説を確かめ、すでにあるものを改善し、価値を生まないことをやめるために、その力を使うこともできる。ただし、自然にそうなるわけではない。解放された時間や資源をどこに使うかを選ばなければならない。

Malorean Corp.のVivienが気づいたのは、昔の仕事が占める時間が減ったということだけではない。成功する保証はなくても、それまでの経験を別の問題に役立てられるということだった。進歩の意味は、自動化できた仕事だけでなく、その先に何が可能になるかにもある。

物語の初め、誰もがLucasの開発の速さに驚いていた。終わりには問いが変わっている。何のために、誰のために、どんな結果を考えて作るのか。

AIの課題は、私たちに際限なく多くの仕事をさせることではない。本当に大切なことを、これまで以上にできるようにすることだ。