DORA

2024

Google Cloud

Accelerate State of DevOps

Gold スポンサー

catchpoint chronosphere DATADOG

Deloitte. Excella Gearset

liatrio Middleware OPSERA

10

DORA の 10 年

目次

エグゼクティブ サマリー

DORA は、パフォーマンスの高いテクノロジー主導型チームおよび組織の能力、手法、尺度について 10 年以上にわたって調査を行ってきました。10 年目となる今回の DORA レポートでは、業種も規模もさまざまな世界中の組織で働く、39,000 人以上のプロフェッショナルの方々の回答を得ました。長期的な取り組みであるこの調査に貴重なご協力をいただき、ありがとうございました。

DORA は、技術職や隣接する職務に従事するプロフェッショナルを対象として世界規模のアンケートを毎年行い、データを収集しています。このアンケートには、組織全体やその組織で働く人に関連した、働き方や成果に関する質問が含まれています。

DORA はそうした要因の相互関係や、それぞれの要因がチームや組織の成功にどのように貢献しているかを把握するために、厳密な統計評価手法を使用しています。

今年は、さらに詳しい分析情報を得て複数の視点から調査結果を掘り下げるために、一部のプロフェッショナルに詳細なインタビューを行い、アンケートを補強しました。詳しくは手法の章をご覧ください。

今年調査した主な成果や結果:

燃え尽き症候群の削減	燃え尽き症候群とは、ストレスが長期間にわたるか過剰であるために起こる、感情的、身体的、精神的な疲労状態です。多くの場合、達成感の欠如、不信感、無関心が見られます。
フロー	フローは、ある人が開発タスクにおいてどれほど集中する傾向があるかを測定します。
仕事の満足度	仕事の満足度では、仕事に対する全体的な感情を測定します。
組織パフォーマンス	収益性、マーケット シェア、顧客総数、業務効率、顧客満足度、プロダクトとサービスの品質、目標達成能力などの分野における組織のパフォーマンスを測定します。
プロダクト パフォーマンス	プロダクトのユーザビリティ、機能性、価値、可用性、パフォーマンス（レイテンシなど）、セキュリティを測定します。
生産性	生産性では、価値の創出やタスクの達成に関して、個人が自分の仕事を効果的、効率的だと感じている度合いを測定します。
チーム パフォーマンス	チームの協調性、革新性、業務効率性、相互信頼性、適応性を測定します。

主な調査結果

幅広い影響を及ぼす AI

AI は、ソフトウェア開発の分野にパラダイムシフトを起こしています。慎重な姿勢が見られるものの、早期に導入した組織では有望な結果が示されています。

AI の導入は次の点で役立ちます。

- フロー
- 生産性
- 仕事の満足度
- コードの品質
- 内部ドキュメント
- レビュー プロセス
- チーム パフォーマンス
- 組織パフォーマンス

しかし、AI の導入には弊害もあります。ソフトウェア デリバリー パフォーマンスの低下が確認されており、プロダクト パフォーマンスへの影響は不明確です。さらに、AI の導入が進むにつれ、価値の高い仕事にかける時間が減っていることが報告されています。この興味深い調査結果については後ほど詳しく説明します。

チームは引き続き試験的な運用を行い、AI への依存度が高まることの影響を詳しく把握する必要があります。

AI への信頼が高まるにつれて AI の導入が増加

生成 AI を活用することで、開発者は生産性の向上を実感し、生成 AI を信頼している開発者はさらに生成 AI を利用するようになります。この点には改善の余地があり、回答者の 39.2% は、AI をあまり、またはまったく信頼していないと回答しています。

ユーザー中心でパフォーマンスが向上

エンドユーザー エクスペリエンスを優先する組織は品質の高いプロダクトを生み出しており、開発者の生産性と満足度が高く、燃え尽き症候群になる可能性が低くなっています。

変革的なリーダーシップの重要性

変革的なリーダーシップは、従業員の生産性、仕事の満足度、チーム パフォーマンス、プロダクト パフォーマンス、組織パフォーマンスを向上させるとともに、従業員の燃え尽き症候群を減らすためにも役立ちます。

優先事項が安定していれば生産性とウェルビーイングが高まる

組織の優先事項が不安定だと、強力なリーダーや優れた内部ドキュメントを備えた組織がユーザー中心アプローチでソフトウェア開発を行ったとしても、生産性が著しく低下し、燃え尽き症候群が大幅に増加します。

プラットフォーム エンジニアリングが生産性向上につながる

プラットフォーム エンジニアリングは生産性と組織パフォーマンスにプラスの影響を与えますが、ソフトウェア デリバリー パフォーマンスに関しては、注意すべき兆候があります。

クラウドでインフラストラクチャに柔軟性が生まれる

柔軟なインフラストラクチャは組織のパフォーマンスを向上させることができます。しかし、クラウドへの移行を、それが実現するはずの柔軟性を取り入れることなく進めた場合、データセンターの利用を続けるよりも有害になる可能性があります。移行を成功させるには、アプローチ、プロセス、テクノロジーの変革が必要です。

高度なソフトウェア デリバリー パフォーマンスを達成可能

パフォーマンスの最も高い部類に入るチームは 4 つのソフトウェア デリバリー指標（変更のリードタイム、デプロイの頻度、変更時の障害率、デプロイ失敗時の復旧までの時間）のすべてで優れている一方、パフォーマンスの最も低い部類に入るチームは 4 つのすべてで劣っています。各パフォーマンス クラスタには、あらゆる業種のチームが含まれています。

DORA の分析情報を当てはめる

DORA でチームと組織の改善を促進するには、現状を評価し、投資すべき分野を特定して改善を行い、フィードバック ループを設けて進捗を把握する必要があります。「継続的な改善」の考え方と手法を採用しているチームは、成功する可能性が高くなります。それを長期的に繰り返すために必要な、組織の「筋力」の構築に投資しましょう。

DORA の調査結果から、試験運用や仮説の立案に役立つ情報を得ることができます。チームや組織にとって最も効果的な方法を確認するために、試験的な運用を行い、変更の影響を測定することが重要です。そうすることでまた、DORA の調査結果を検証できます。異なる結果が出ることもありますが、皆様の経験を今後に活かせるよう、進捗状況を共有していただければ幸いです。

改善には実験的なアプローチを取ることをおすすめします。

1. 改善する分野や結果を明確にします。
2. ベースラインまたは現状を測定します。
3. 望ましい状態に近づくために必要な対処の仮説を立てます。
4. 意見をまとめて改善計画を固めます。
5. 取り組みを実行します。
6. 進捗を測定します。
7. プロセスを繰り返します。改善は繰り返しによって徐々に達成されます。

DORA COMMUNITY

1 人では改善できません

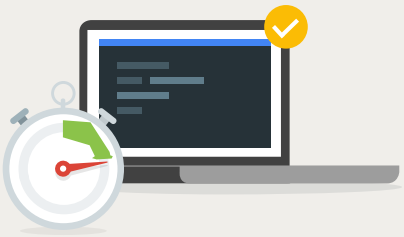
互いの経験から学びましょう。DORA コミュニティ（**https://dora.community**）は、改善の取り組みについて共有し学ぶことができるフォーラムです。

ソフトウェア デリバリー パフォーマンス

テクノロジー主導型チームには、現状を評価し、改善の優先順位を決め、進捗を検証するために、パフォーマンスを測定する方法が必要です。DORA は、ソフトウェア デリバリー プロセスの成果を効果的に測定できる 4 つのソフトウェア デリバリー指標（4 つの主要指標）を繰り返し検証してきました。

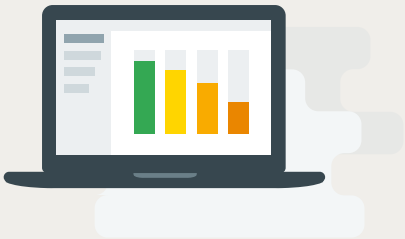
4 つの主要指標

DORA の 4 つの主要指標は、ソフトウェアの変更におけるスループットと安定性を測定するために使用されてきました。これには、構成の変更やコードの変更など、あらゆる変更が含まれます。

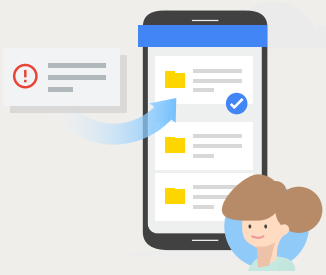


変更のリードタイム:
コードの commit または変更が本番環境へ適切にデプロイされるまでの時間。

デプロイの頻度:
アプリケーションの変更を本番環境にデプロイする頻度。



変更時の障害率:
デプロイが原因で本番環境で障害が発生する割合（ホットフィックスやロールバックが必要）[1]。



デプロイ失敗時の復旧までの時間:
デプロイの失敗時に復旧にかかる時間。

通常これらの指標は連動しており、パフォーマンスの最も高い部類に入るチームは 4 つの指標すべてで優れ、パフォーマンスの最も低い部類に入るチームは 4 つのすべてで劣っていました。

ソフトウェア デリバリー パフォーマンスの尺度の改良

4 つの主要指標の分析では、これまで変更時の障害率が外れ値となっていました[2]。変更時の障害率は他の 3 つの指標と強い相関があるものの、統計のテストと手法により、これら 4 つを 1 つの要素にまとめることはできません。変更時の障害率に関する質問への回答方法を変更したことで、関係の改善が見られましたが、他にも何かが起きていると思われました。

変更時の障害率という指標は、チームに求められるやり直し作業の量に相当する、というのがこれまでの仮説でした。デリバリーに失敗すると、チームはその変更を修正するために、たいていは別の変更を導入する必要があります。

この仮説を確かめるために、今年、アプリケーションのやり直し率に関する以下の質問を新たに追加しました。「主なアプリケーションまたはサービスに関して、過去 6 か月間に、ユーザーが直面しているバグに対処するために、計画はされていなかったが実施したデプロイはどれくらいありましたか。」

データを分析したところ、やり直し率と変更時の障害率は関連するという仮説が確認されました。この 2 つの指標を組み合わせれば、ソフトウェア デリバリーの安定性に関する信頼性の高い要素となります。

これは、ソフトウェア パフォーマンス レベルの分析にも表れています。今年の調査では、チームの半数以上でソフトウェアのスループットとソフトウェアの安定性に違いが見られました。こうした違いから、ソフトウェア デリバリー パフォーマンスを 2 種類の要素に基づいて考えるようになりました。

コンセプト	
ソフトウェア デリバリー パフォーマンス	
要素	
ソフトウェア デリバリーのスループット	**ソフトウェア デリバリーの安定性**
使用する指標	
• **変更のリードタイム** • **デプロイの頻度** • **デプロイ失敗時の復旧までの時間**	• **変更時の障害率** • **やり直し率**

本レポートの全体を通して、DORA の分析では、ソフトウェア デリバリー パフォーマンスのコンセプトと両方の要素がさまざまな機会に活用されています。また、ソフトウェア デリバリー パフォーマンスを表すにあたり、5 つの指標がすべて考慮されています。

変更のリードタイム、デプロイの頻度、デプロイ失敗時の復旧までの時間は、ソフトウェア デリバリー スループットを表す際に使用します。この要素は、あらゆる更新、通常の変更、障害対応時の変更の速度を測定します。

変更時の障害率とやり直し率は、ソフトウェア デリバリーの安定性を表す際に使用します。この要素は、意図せずデプロイの後ですぐに追加作業が必要になる可能性を測定します。

パフォーマンス レベル

DORA では毎年、アンケート回答者に、彼らが取り組んでいる主要なアプリケーションまたはサービスのソフトウェア デリバリーのパフォーマンスについて質問しています。得られた回答は、クラスタ分析を使用して分析します。これは、互いに類似しているが他の回答グループとは異なる回答を識別する統計的手法です。

過去のクラスタ分析との整合性を維持するため、元の 4 つのソフトウェア デリバリー指標についてクラスタ分析を実施しました。

ソフトウェア デリバリー パフォーマンスの分析において、回答のクラスタが 4 つ表れました。これらのレベルは事前に設定するのではなく、アンケートの回答から導き出すようにしています。そうすることで、毎年、回答者全体のソフトウェア デリバリー パフォーマンスの概要を把握できます。

今年はデータから以下に示す 4 種類のクラスタが表れました。

パフォーマンス レベル	変更の リードタイム	デプロイの頻度	変更時の 障害率	デプロイ失敗時の 復旧までの時間	回答者の割合*
エリート	1 日未満	必要に応じて（1 日に複数回のデプロイ）	5%	1 時間未満	19%（18〜20%）
高	1 日から 1 週間の間	1 日 1 回から週 1 回の間	20%	1 日未満	22%（21〜23%）
中	1 週間から 1 か月の間	週 1 回から月 1 回の間	10%	1 日未満	35%（33〜36%）
低	1 か月から 6 か月の間	月 1 回から 6 か月に 1 回の間	40%	1 週間から 1 か月の間	25%（23〜26%）

* 89% 不確実性区間

スループットか安定性か

4 つのクラスタすべてで、スループットと安定性が相関しています。この相関は、高パフォーマンス クラスタ（黄色）よりスループットが低く安定性が高い、中パフォーマンス クラスタ（橙色）でも見られます。ここから、スループットと安定性以外の要素がクラスタのパフォーマンスに影響していることが伺えます。たとえば中パフォーマンス クラスタは、変更の提供頻度を高めることが有益である可能性があります。

デプロイの頻度を高めるのと、デプロイ時の失敗を減らすのでは、どちらが良いでしょうか。

この質問に対する普遍的な答えはないでしょう。対象のアプリケーションやサービス、そのアプリケーションに取り組んでいるチームの目標、さらには最も重要な点として、アプリケーションのユーザーが期待している内容によって異なります。

DORA では、あえて、迅速なチームを「高パフォーマー」、遅くても安定しているチームを「中パフォーマー」と呼ぶことにしました。この決定は、このようなパフォーマンス レベルを使用する際に注意すべき点を浮き彫りにします。つまりチームにとっては、特定のパフォーマンス レベルに達することよりも、改善することのほうが重要なはずだということです。最高のチームは、卓越した**改善**を達成するチームであって、必ずしも卓越した**パフォーマンス**を発揮するチームではありません。

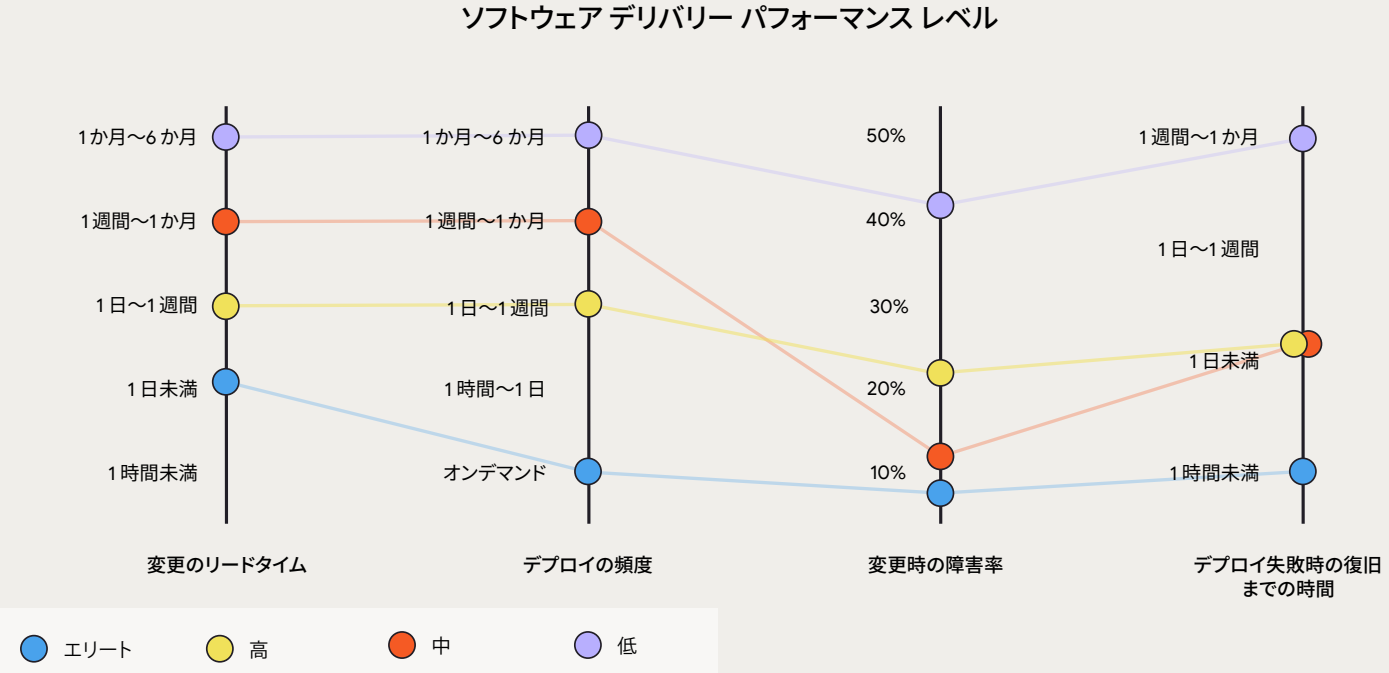


図 1: ソフトウェア デリバリー パフォーマンス レベル

低パフォーマーと比較した場合のエリート パフォーマーの実績

127 倍	182 倍	8 倍	2,293 倍
リードタイムが短い	1 年あたりのデプロイ件数	変更時の障害率が低い	デプロイ失敗時の復旧までの時間が短い

パフォーマンス クラスタの使い方

パフォーマンス別のクラスタ分けによって、今年のアンケート回答者のソフトウェア デリバリーのパフォーマンスが表れているベンチマーク データを導出できます。クラスタ分析は、エリート パフォーマンスが実現可能であることをすべての人に理解してもらうことを目的としています。

特定のパフォーマンス レベルに達することよりも重要なのは、チームが全体的なパフォーマンスの改善に注力することだと、DORA では考えています。最高のチームは、卓越した**改善**を達成するチームであって、必ずしも卓越した**パフォーマンス**を発揮するチームではありません。

業界はパフォーマンス レベルに意味のある影響を与えない

DORA の調査において、業界がソフトウェア デリバリー パフォーマンスの予測因子となることはほとんどありません[3]。高いパフォーマンスを発揮するチームは、あらゆる業種で確認されています。これは、さまざまな業界に固有の課題がないことを示しているわけではありませんが、ソフトウェア デリバリー パフォーマンスに関しては、どの業界にも、特有の障害または強みがあるようには見えません。

ソフトウェア デリバリー パフォーマンス指標を使用する

アプリケーションやサービスには、それぞれ独自のコンテキストがあります。この複雑さにより、ある変更がシステム全体のパフォーマンスにどのような影響を与えるかを予測することが難しくなっています。さらに、組織で一度に 1 つのことだけを変更するのはほぼ不可能です。この複雑さを念頭に置き、ソフトウェア デリバリー パフォーマンス指標をどのように使用すれば、改善の取り組みの指針として役立つでしょうか。

まず、測定と改善を行う主なアプリケーションやサービスを特定します。そのアプリケーションを担当する部門横断的なチームを編成し、現在のソフトウェア デリバリー パフォーマンスを測定して共通の理解を形成することをおすすめします。DORA クイック チェック（https://dora.dev/quickcheck）を使用すると、会話を進め、ベースラインの指標を定めることができます。チームは、パフォーマンスの改善の妨げとなっているものを把握する必要があります。

そうした障害を見つける効果的な方法として、チームでバリュー ストリーム マッピング演習を行うことが挙げられます[4]。

次に、改善計画を明らかにし、合意します。この計画では、DORA で調査した多くの能力のうちの一つを改善することに集中しても構いません[5]。また、アプリケーションや組織に固有な他のものでも構いません。

計画を用意できたら、行動に移します。改善の取り組みに力を尽くし、その過程で得た教訓に注目しましょう。

変更が実施され、定着したら、今度は 4 つの主要指標を再評価します。変更が実施された後、チームはどのように変わりましたか。どのような教訓が得られましたか。

このプロセスを繰り返すことで、チームに継続的な改善の手法が定着します。

変化は一朝一夕に起こるものではありません。学習、迅速なフロー、迅速なフィードバックを実現する反復的なアプローチが必要です[6]。

1. デプロイが本番環境導入後に問題を引き起こし、エンドユーザーがその問題を経験する可能性がある場合にのみ、そのデプロイで変更時の障害が発生したと見なします。対照的に、本番環境に導入する途中で変更が中止された場合は、デプロイ プロセスにおけるエラーの検出が適切に機能したと捉えます。
2. Forsgren, Nicole, Jez Humble, and Gene Kim.2018.Accelerate: The Science Behind DevOps : Building and Scaling High Performing Technology Organizations.IT Revolution Press. pp. 37-38
3. 2019 年の Accelerate State of DevOps レポート（p 32）によると、小売業界ではソフトウェア デリバリー パフォーマンスが大幅に高くなっています。https://dora.dev/research/2019/dora-report/2019-dora-accelerate-state-of-devops-report.pdf#page=32
4. https://dora.dev/guides/value-stream-management/
5. https://dora.dev/capabilities
6. https://dora.dev/research

AI: 導入と考え方

要点

アンケートの対象となった全業界の大多数の組織が、自社のアプリケーションやサービスに AI を深く組み込む方向へ軸足を移しています。開発担当者の大半も、中核的な職務を遂行するにあたって AI を活用しており、その結果、生産性の向上が報告されています。現在の市場で競争力を維持するには AI を利用する必要があるという認識が開発担当者の間で広く浸透しており、これが組織にとっても個々の開発担当者にとっても、AI 導入の重要な推進力となっているようです。

はじめに

AI の影響を概説する一般的なニュース記事は、良いもの[1]、悪いもの[2]、そして酷いもの[3]などさまざまでしたが、その拡散状況を考えると、今年 AI が開発作業の状況に及ぼした大きな影響を無視することは難しいでしょう。そこで、2023 年の Accelerate State of DevOps Report では AI をパフォーマンスに影響する数多くの技術的能力の一つとして論じたにすぎませんでしたが[4]、今年はこのトピックをさらに詳しく取り上げます。

専門的な開発作業における AI の利用が初歩的な段階から普及段階へと急速に移行する中、2024 年の Accelerate State of DevOps Report は、業界の重要な転換点において、開発担当者による導入、利用、考え方を評価する重要な機会となるでしょう。

調査結果

AI の導入

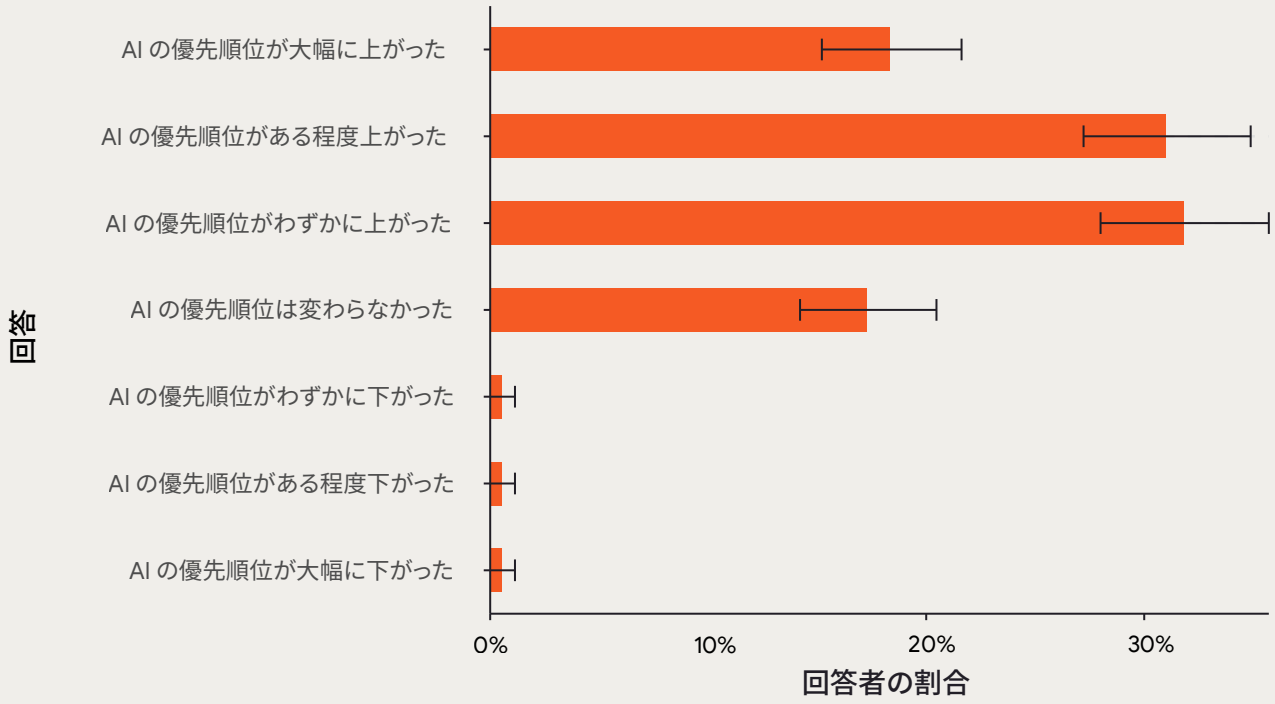
AI の導入に関する調査結果は、AI がもはや「差し迫っている」のではなく、完全に到来し、おそらく今後も存在し続けるとの認識が広がっていることを示しています。

AI の組織的な導入

回答者の大多数（81%）が、自社のアプリケーションやサービスへの AI の組み込みを増やすように、組織が優先度を変えたと回答しています。回答者の49.2%は、この変更の規模を「ある程度」または「大幅」と表現しています。

所属組織の AI への注力が減っているとした回答者が 3% だったことは注目に値します。これはアンケートの誤差の範囲内です。回答者の 78% は、こうした優先度の変化の結果、AI をどのように利用する予定なのかについて組織が透明性を確保していると思うと回答しています。このデータを図 2 に示します。

AI に関する組織の優先度の変化



エラーバーは 89% 不確実性区間を表す

図 2: 自社のアプリケーションやサービスに AI を組み込む優先度の変化に対する回答者の認識

アンケートの対象となったどの業界の回答者も、日常の業務で AI を利用していると回答した割合は統計上同じ水準でした。この点から示唆されるのは、AI のこのような急速な導入が、あらゆる産業分野にわたって満遍なく進展しているということです。これは、少し意外な結果でした。規制によって課せられている制約のレベルや、これまでの技術革新のペースは業界によって千差万別であり、それぞれの差異がテクノロジーの導入率に影響を与えるはずだからです。

しかし、大規模な組織で働く回答者は、小規模な組織で働く回答者よりも日常業務における AI への依存度が低いことがわかりました。これは、大規模な組織では組織の複雑さと調整の費用が増すため、テクノロジーの変化への対応が遅くなるとした先行文献と一致しています[5]。

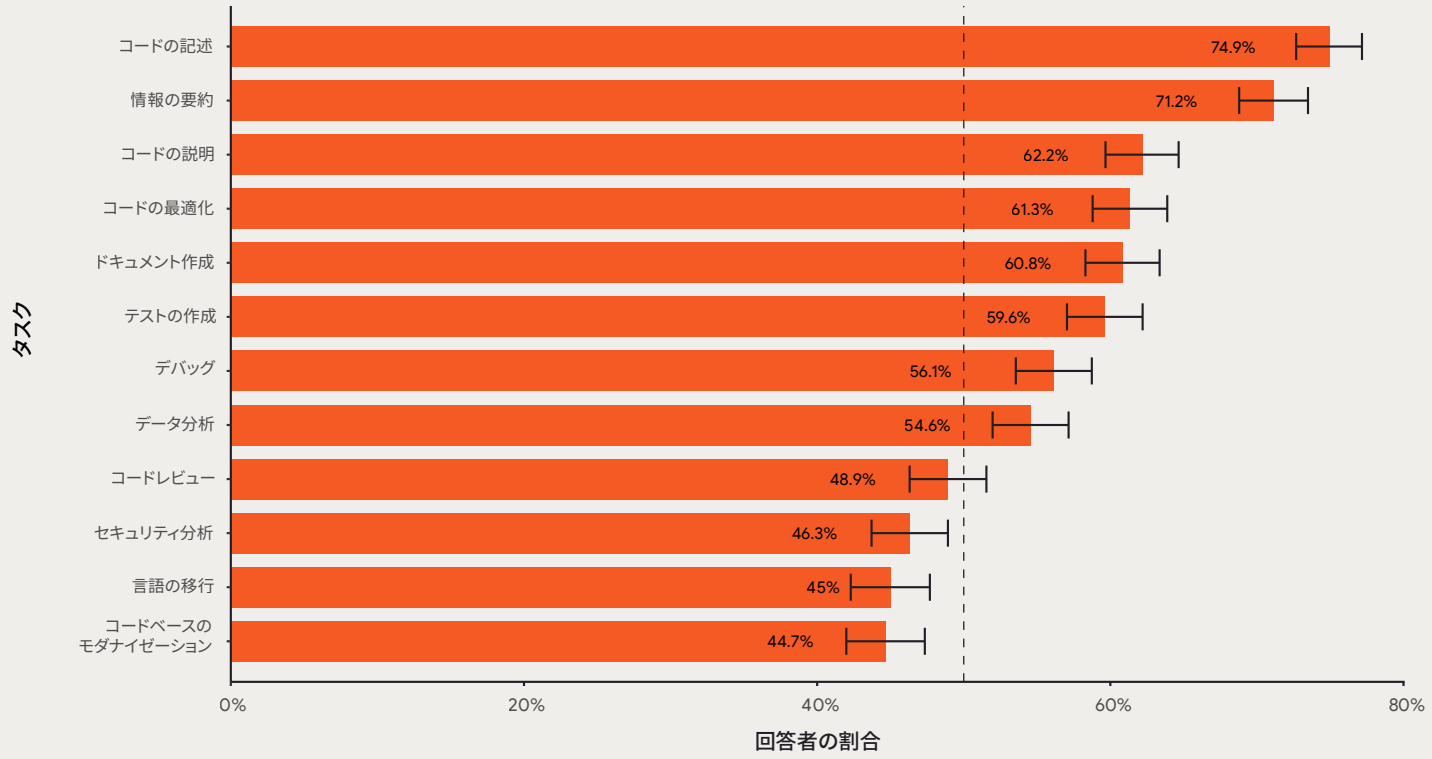
AI の個人的な導入

個人レベルでは、回答者の 75.9% が、1 つ以上の日常業務において少なくとも部分的には AI を利用していることがわかりました。職務に以下のタスクが含まれる回答者の大半が AI を利用しています。

1. コードの記述
2. 情報の要約
3. 不明確なコードの説明
4. コードの最適化
5. コードに関するドキュメント作成
6. テストの作成
7. コードのデバッグ
8. データ分析

アンケートの回答に含まれていたすべてのタスクの中で、ソフトウェア開発の業務における最も一般的な AI のユースケースは、コードの作成と情報の要約でした。職務にこれらのタスクが含まれていて、そのタスクの少なくとも一部で AI を利用しているとした回答者は、それぞれ 74.9% と 71.2% です。このデータを図 3 に示します。

AI を利用するタスク



エラーバーは 89% 信頼区間を表す

図 3: 12 種類の一般的な開発タスクを実施する際に少なくとも部分的には AI を利用する回答者の割合

回答者が日常業務で AI を扱う際に最もよく利用するインターフェースは chatbot（78.2%）で、次いで外部ウェブ インターフェース（73.9%）、IDE に組み込まれた AI ツール（72.9%）でした。回答者が内部ウェブ インターフェース（58.1%）や自動 CI / CD パイプライン（50.2%）の一部で AI を利用するケースは、比較的少数派でした。

ただし、CI / CD パイプラインや内部プラットフォームで AI が利用されていることを回答者が認識しているかどうかは、そうしたテクノロジーを扱う頻度に依存する可能性が高いと考えられます。そのため、これらの数値は人為的に低くなっている可能性があります。

データ サイエンティストや ML の専門家は、他のどの職種の回答者よりも、AI を利用する傾向が高いことがわかりました。逆にハードウェア エンジニアは、他のどの職種の回答者よりも、AI を利用する傾向が低くなっています。これは、ハードウェア エンジニアの職務が、AI がよく利用されている前述のタスクとは異なっているためだと思われます。

AI 導入の推進力

インタビュー参加者は、AI を導入するという決定を、競争上のプレッシャーや、組織と開発者の双方が業界標準に後れを取らないようにする必要があることに関連付けるケースが多く、AI に習熟することへの関心がますます高まっています。

複数の参加者の組織において、AI の活用が「マーケティングにおける強み」（P3）となり[6]、競合他社との差別化に役立つと考えられていました。競合他社がプロセスに AI を導入し始めていると知ったことで、ある会社は新しいテクノロジーの導入に伴って通常は踏んでいた「膨大で煩雑な手続き」を割愛しました。これは、「競合他社が先にそのような行動を取ったらどうなるか」という問いから、AI の導入に緊急性を感じたためです（P11）。

個人レベルでは多くの参加者が、ソフトウェア開発において AI の利用に習熟することは「エンジニアとしての新たな参入障壁のようなもの」だという考え方に AI の導入を結びつけていました（P9）。複数の参加者が他の開発者に対し、開発ワークフローに AI を迅速に導入すべきだと提案しています。その理由として「この分野ではたくさんのことが起きていて、ついていくのがやっとです。利用しないとすぐに取り残されることになると思います」（P4）とのコメントがあります。

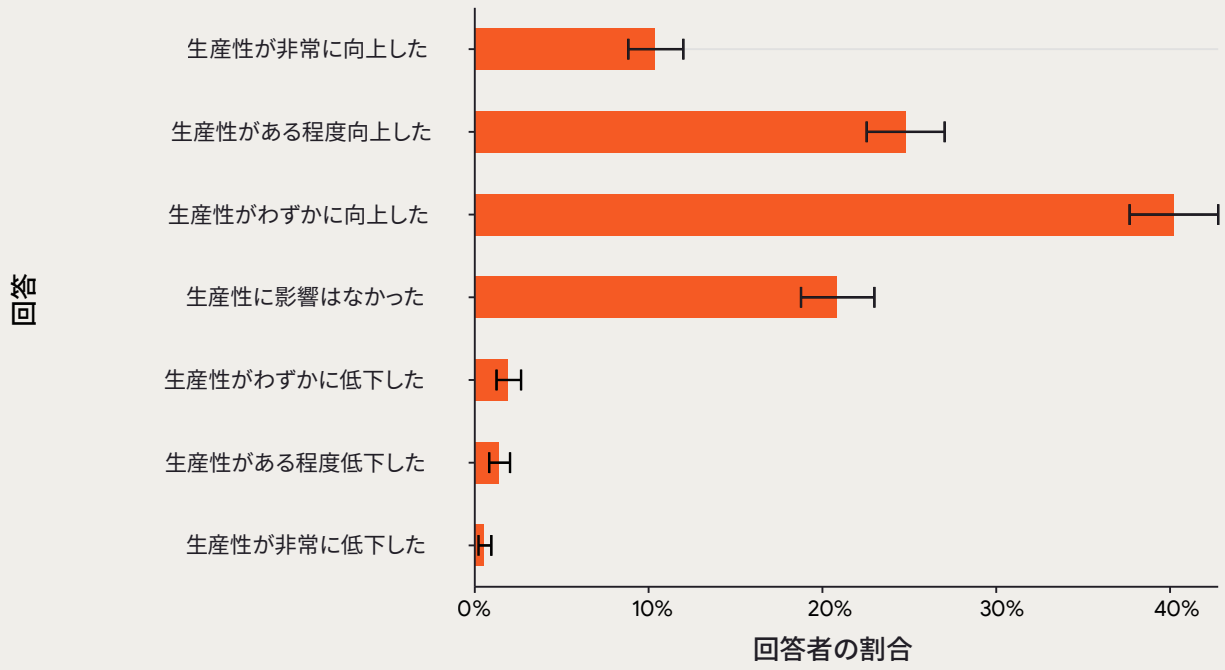
AI に対する認識

AI によるパフォーマンスの向上

AI を導入している多くの組織や開発者にとって、開発業務で AI を利用することのメリットはかなり大きいようです。2024 年の初めに実施したアンケートにおいて、回答者の 75% が直前の 3 か月間に AI で生産性が向上したと回答しています。

注目すべきは、回答者の 3 分の 1 以上が、ある程度（25%）または非常に大きな生産性の向上（10%）が確認されたと表現していることです。AI によって生産性にわずかでも悪影響があったとした回答者は 10% 未満でした。このデータを図 4 に示します。

AI による生産性の変化に対する認識



エラーバーは 89% 不確実性区間を表す

図 4: AI が生産性に与える影響についての回答者の認識

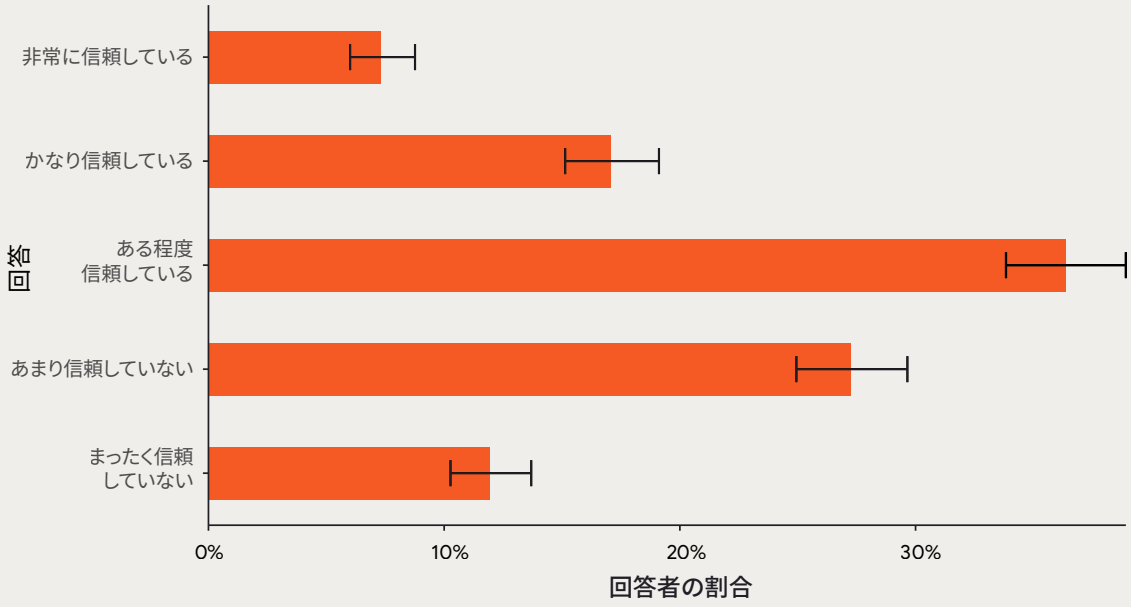
さまざまな職務のうち、AI による生産性の向上を特に大きく回答したのは、セキュリティ担当者、システム管理者、フルスタック開発者でした。モバイル開発者、サイト信頼性エンジニア、プロジェクト マネージャーも生産性の向上について肯定的な回答をしていますが、回答した生産性向上の規模は他のどの職務の回答者よりも小さくなっています。

AI が開発業務で利用されるようになってからはまだ日が浅く、ユーザーの習熟度もそれに応じた水準になります。したがって、コードの作成ではデベロッパーの実務の妨げになっているのではないかと考えられましたが、調査結果はその仮説に反するものでした。程度を問わず、AI がコードの作成の妨げになっていると述べた回答者は 5% にすぎません。具体的には、AI を活用したコーディング ツールを導入した結果、コードの作成実務で少なくともなんらかの改善があったと述べたのは、回答者の 67% にのぼっています。約 10% は、AI によってコードの作成実務で「非常に」大きな改善があったと述べています。

AI 生成のコードへの信頼度

開発業務で使用する AI 生成のコードの信頼性に対する参加者の認識は、複雑でした。回答者の大半（87.9%）は、AI 生成のコードの品質をある程度信頼していると回答していますが、AI 生成のコードの品質に対する回答者の信頼度は概して低く、39.2% は「あまり信頼していない」（27.3%）か、「まったく信頼していない」（11.9%）と回答しています。このデータを図 5 に示します。

AI 生成のコードの品質に対する信頼度



エラーバーは 89% 不確実性区間を表す

図 5: 回答者が報告した AI 生成のコードの品質に対する信頼度

開発者が急速に AI を導入して利用し、パフォーマンスの向上に貢献するものとして認識しているというアンケート結果を考えると、全体的に AI が信頼されていないことは意外でした。インタビューにおいて多くの参加者が、業務で使用する AI 生成のコードの出力を調整することに前向きである、またはそうするつもりであるとしたことは注目に値します。

AI 生成のコードの出力を評価し修正する必要性について、ある参加者は「初期の StackOverflow」に例え、「あの頃は、StackOverflow のユーザーは本当に経験豊富で、すべきことが正確にわかっているのだと思っていました。そして、単にコピーして貼り付けていたら、とんでもないことになったものです」と述べています（P2）。

これは新しい問題ではないこともあり、P3 のような参加者は、既存のコードの品質保証プロセスについて会社は「何層ものチェックがあるため、誰かが単に Copilot や ChatGPT からコードをコピーして貼り付けることについて心配はしていない」と感じています。

AI 生成のコードの正確性に関して開発者は必ずしも絶対的な信頼を期待しておらず、絶対的な信頼は、AI 生成のコードが有用であると開発者が判断するうえで必須というわけでもない、という仮説を立てました。むしろ、おおむね正しい AI 生成のコードが多少の手直しで申し分なく仕上がるのであれば許容され、幅広い導入と利用を促すうえでは十分に価値があり、既存の品質保証プロセスとも共存できると考えられます。

今後の AI に関する予想

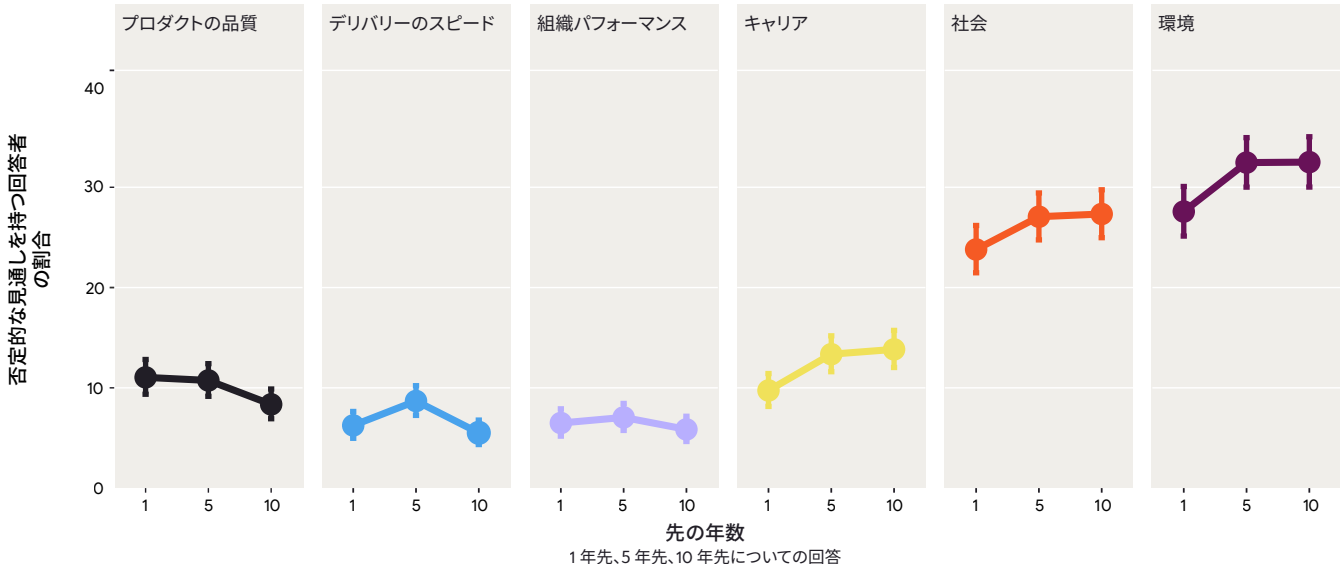
全体的に、調査結果は AI がすでに開発担当者の業務に大きな影響を与えていることを示しており、この傾向は今後も拡大していくと考えられます。将来的に AI が開発や日常生活にどのような影響を与えるようになるかを正確に予測することは不可能ですが、回答者には、今後の 1 年、5 年、10 年で AI がもたらす影響について推測して予想を共有してもらいました。

回答者は最近の経験に基づき、AI が開発業務に与える影響についてかなり肯定的な回答をしていますが、AI の将来的な影響についての予測は、あまり希望に満ちたものではありませんでした。

楽観的な面を見ると、回答者は、今後 1 年、5 年、10 年で AI によってプロダクトの品質は向上し続けると予想しています。これは、AI が開発担当者のパフォーマンスにプラスの影響を与えているという調査結果と一致しています。

しかし、AI がキャリア、環境、社会全体に根本的な悪影響を与え、そうした悪影響は約 5 年で完全に現実化するとも予想しています。このデータを図 6 に示します。

予想される AI の悪影響



エラーバーは 89% 信頼区間を表す

図 6: 今後の 1 年、5 年、10 年で AI がもたらす悪影響についての回答者の予想

インタビュー参加者は将来的な AI の影響について、アンケート回答者と同様に複雑な感情を抱いていました。規制状況がはっきりしない状況で将来の法的措置について考えを巡らせ、「はっきりしたときに不利な立場になる」のではないかと心配する回答者もいました（P3）。

長年にわたって人々が抱いてきた不安をなぞり、「AI は人間の代わりになるのでしょうか。誰もわかりません。なるかもしれません」と言う人もいました（P2）。一方、他の人はその不安を否定して過去の事例を引き合いに出し、「『2000 年問題で何もかも破滅する』などと言っていた人もいました。当時は目新しいことだったから。

（けれども）置き換わるようなことは何もなく、実際には多くの雇用が生み出されました。AI でも同じことが起こると思います」と述べました（P1）。

将来的に AI がもたらす影響は、依然として不明です。しかし今年のアンケートでは、AI がソフトウェア開発分野において無視できないほどのパラダイム シフトを引き起こしていることが、強く示されました。これまでのところ、この変化は開発担当者に好意的に受け止められています。

1. https://www.sciencedaily.com/releases/2024/03/240306144729.htm
2. https://tech.co/news/list-ai-failures-mistakes-errors
3. https://klyker.com/absurd-yoga-poses-generated-by-ai/
4. https://dora.dev/dora-report-2023
5. Rogers, Everett M., Arvind Singhal, and Margaret M. Quinlan.“Diffusion of innovations.”An integrated approach to communication theory and research.Routledge, 2014.432-44, Tornatzky, L. G., & Fleischer, M. (1990).The processes of technological innovation. Lexington, MA: Lexington Books
6. (P1)など、(P[N]) は、インタビュー参加者の仮名を表しています。

AI がダウンストリームに与える影響を探る

要点

この章では、個々の開発者から組織全体に至るまで、AI の導入が及ぼす影響を幅広い観点から調査します。調査結果からは、明らかな利点と予期せぬ欠点の両方を含む、複雑な状況が明らかになりました。AI の導入により、個々の生産性、フロー、仕事の満足度は高まる一方、価値の高い仕事にかける時間が短くなる可能性があります。

同様に、AI はコードの品質、ドキュメント、レビュー プロセスにプラスの影響を与えますが、驚くべきことに、そうした利点はソフトウェア デリバリー パフォーマンスの向上にはつながりません。むしろ AI の導入はこの分野では有害であるように見えますが、プロダクト パフォーマンスへの影響は無視できる程度にとどまっています。

こうした課題があるにもかかわらず、AI の導入はチームや組織のパフォーマンスの向上に結びついています。この章では最後に、ソフトウェア開発における AI の役割を批判的に評価することと、利点を最大限に高めて予期せぬ結果を軽減するために、積極的に AI の使い方を工夫することを呼びかけます。

AI の現在と DORA

大手テクノロジー企業は今後 5 年間で AI の開発に約 1 兆ドルを投資すると試算されています[1]。これは「AI: 導入と考え方」の章で紹介した、回答者の 81% は会社が AI の開発にリソースをシフトしているとした統計とよく一致しています。

AI の環境への影響は、さらに費用を増大させます。2030 年までに AI によってデータセンターの電力需要が 160% 増加するとの試算もあります[2]。1 つの AI モデルのトレーニングに、おおむね「米国の 1,000 世帯以上の年間電力需要」の電力が必要になる可能性があります[3]。回答者の 30% 以上が AI は環境に悪影響を及ぼすと考えているのも頷けます。

開発や環境に関するコストの他に、導入に伴うコストが発生する可能性もあります。

これは、生産性の低下や専門家の雇用など、さまざまな形で表れるでしょう。こうした導入コストが社会レベルで発生することも考えられます。回答者の 3 分の 1 以上は、今後 10 年間で AI が社会に悪影響を及ぼすと考えています。このようなコストを踏まえると、得られる結果について大きな関心が集まるのは当然のことに思えます。

そうした関心は多様なメディア、記事、調査に表れており、少なくともある程度は異なる感情、データが入り交じっています。

AI が人類の能力を劇的に向上させたと考える人もいれば[4]、AI は宿題を手伝ってくれる程度のツールにすぎないと言う人や[5]、AI が人類の凋落につながることを懸念する人もいます[6]。

特定のタスクを適切に遂行する能力など、短期間に表れる成果を示す指標はおおむね肯定的です[7]。チームのコードベースなど、時間がかかる成果では、結果があまり明確ではなくなり、肯定的でもなくなってきます。たとえば一部の調査では、2021 年以前のベースラインと比較してコードチャーンが 2 倍になる可能性があることが示唆されています[8]。

このようなダウンストリームへの影響を把握することが難しいのは、当然のことです。影響が原因から遠くなるほど、関係性は薄れ、明確でなくなります。

AI がダウンストリームに及ぼす影響を評価することは、湖に投げ込まれた石の影響を定量化するようなものです。石が水に当たった点に最も近い波紋は非常に簡単に結びつけられますが、入水した点から離れるほど、石が及ぼした影響は目立たなくなり、石の影響による波だと特定することは難しくなります。

本質的に、AI は他のプロセスやダイナミクスの荒波に投げ込まれた石のようなものです。AI（またはなんらかの技術や手法）によって引き起こされる波の規模を把握するのは困難です。こうしたことも、AI の影響を把握するにあたり、原則に沿った測定と分析のフレームワークを採用することに業界が苦労している理由の一つなのかもしれません[9]。

DORA のアプローチは、特にこうした課題に役立つように設計されています。DORA のリサーチは、ある手法が有用かどうかを判断できるように設計されています。過去 10 年にわたり、セキュリティ対策、変革的なリーダーシップ、創造的文化、ドキュメント作成、継続的インテグレーション、継続的デリバリー、ユーザー中心など、数多くの手法がダウンストリームに及ぼす影響を探ってきました[10]。

DORA のアプローチは[11]、多くの成果にわたって AI が及ぼす作用を探る際には特に、AI の影響を把握するために役立ちます。

AI の導入状況を測定する

AI の導入による影響を把握する際にまず課題となるのは、AI の導入状況の測定です。開発ワークフローにおける AI の重要性を把握するうえで、利用頻度の測定は依存度の測定ほどには有意義ではないだろうと判断しました。コードレビューやドキュメント作成を月に数回か数か月に 1 回行う程度でも、こうしたタスクは業務上非常に重要だと考えることはできます。

逆に、AI を頻繁に利用しているからといって、職務上重要な業務や中心的な業務に AI を利用しているとは限りません。

そうしたことを踏まえ、AI への依存度について、全般的に、また特定のタスクに関連させて回答者に尋ねました。アンケートの結果とその解釈については前の章で詳述しています。

因子分析から、「全般的」な AI への依存度のアンケート項目は、以下のタスクに関して報告された AI への依存度とかなり重複していることがわかりました。

- コードの記述
- 情報の要約
- コードの説明
- コードの最適化
- ドキュメント作成
- テストの作成

この 7 つの項目に見られる強い共通性と共分散は、AI の導入という因子が基となっていることを示唆しています。

AI が個人に及ぼす影響は明確な利点（と潜在的なトレードオフ）

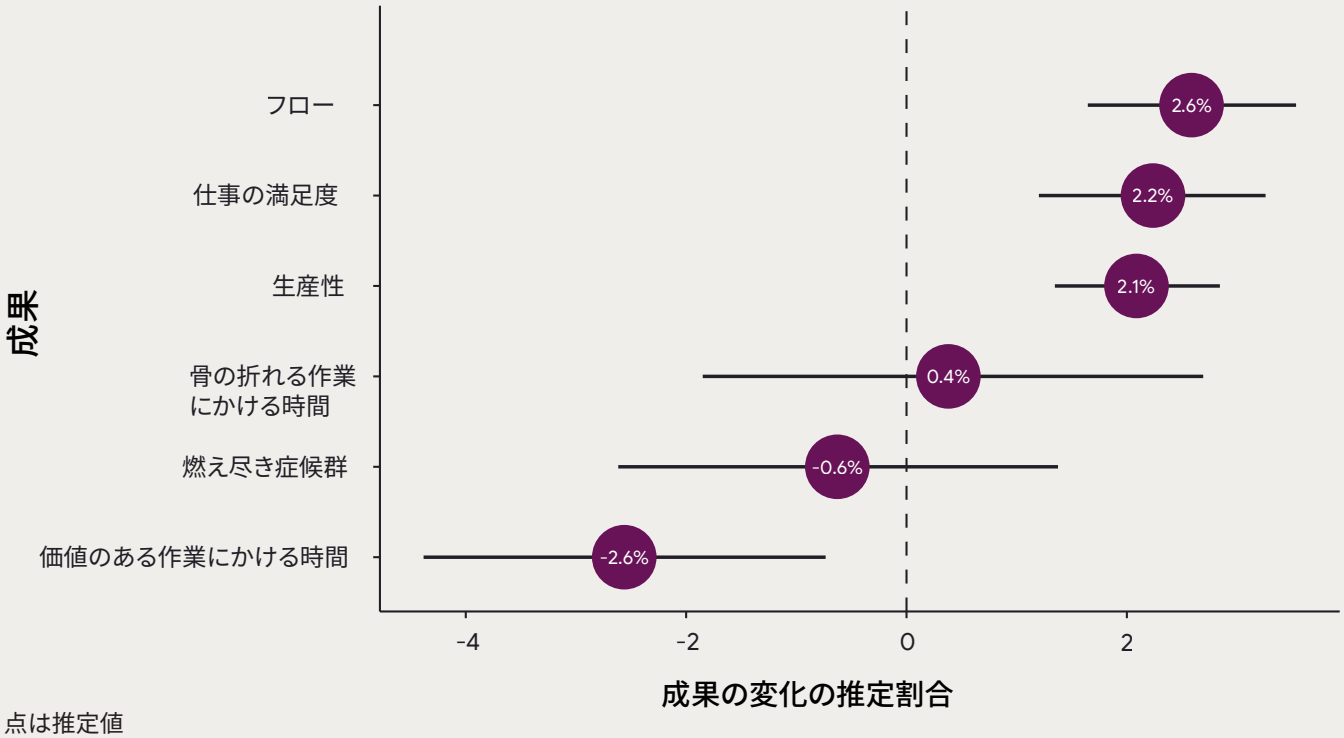
例年どおり、個人の成功とウェルビーイングに関連するさまざまな構成概念を測定しました。

仕事の満足度	仕事に対する個人の全般的な感覚を把握するための単一の項目。
燃え尽き症候群	燃え尽き症候群の身体的、感情的、心理的側面や、私生活への影響など、多面的な性質をまとめた因子。
フロー	開発タスクにおいて個人がどの程度集中する傾向があるかを把握するための単一の項目。
生産性	個人が業務において、価値の創出やタスクの達成に関して自分がどの程度効果的、効率的であると感じているかを測定するための因子スコア。
骨の折れる作業にかける時間	個人が長期的な価値のほとんどない反復的な手動のタスクにかける時間の割合を測定する単一の項目。
価値のある作業にかける時間	個人が価値があると思うタスクにかける時間の割合を測定する単一の項目。

AI の導入に伴って、これらの質問への回答が変化するかどうかを把握しようと考えました。結果は、多くの場合そのとおりであることを示しています。

図 7 に、AI の導入が個人の成功やウェルビーイングに与える影響に関する最善の推定値を示します。

個人の AI の導入が 25% 増えた場合



点は推定値

エラーバーは 89% 不確実性区間

図 7: AI の導入が個人の成功やウェルビーイングに与える影響

明確な利点

個人にとっての AI 導入の利点に関する話はおおむね好意的なものですが、どれほど良い話にも欠点はあるものです。**はっきりしているのは、フロー、生産性、仕事の満足度に AI が実質的かつ有益な影響を与えるということです（図 7 参照）。**

たとえば、個人の AI 導入が 25% 増えると、生産性は約 2.1% 向上すると考えられます（図 7 参照）。小さなことのように思えますが、これは個人レベルの話です。このパターンが数十人、あるいは数万人の開発者に広がっていったら、と想像してみましょう。

このパターンは予想どおりでした。AI がさまざまな情報源を統合し、高度にパーソナライズされた回答を一か所で提供できるからこのような結果になったという面もあるでしょう。これを独力で行うと時間がかかり、状況の切り替えが多く、フローはあまり促進されません。

生産性とフローが仕事の満足度と強い関係を持っていることを考えれば、AI の導入が仕事の満足度の向上につながることは、驚くようなことではありません。

潜在的なトレードオフ

ここから話は少し複雑になります。AI の導入の価値提案として、価値のある作業にかける時間を増やせるというものがあります。つまり、反復的な骨の折れる手動のタスクを自動化することで、回答者が「より良いこと」に時間を自由に使えるようになると想定しています。しかしデータによれば、AI の導入が進むと価値のある作業にかける時間が減るという逆の効果が生じる可能性がある一方で、骨の折れる作業にかける時間は影響を受けないようです。

フロー、仕事の満足度、生産性といった回答者のウェルビーイングの指標は、これまで、価値のある作業にかける時間と関連付けられてきました。そのため、価値のある作業にかける時間の減少とは無関係に、そうした指標で増加が見られたことは驚くべきことです。

こうしたパターンをうまく説明するには、この矛盾のようなものに立ち向かう必要があります。映画をうまく説明する場合、その説明と矛盾するシーンを無視するわけにはいきません。本をうまく説明する場合、その説明に当てはまらない章を無視するわけにはいきません。同様に、こうしたパターンをうまく説明するには、単純なストーリーを伝えることができるからといって、パターンの一部だけに注目するわけにはいきません。

データに当てはまる仮説は無数に考えられますが、妥当と思われる仮説に着目しました。フロー、生産性、仕事の満足度には AI のプラスの影響がある一方、価値のある作業にかける時間は減少し、労力は変わらない、という仮説です。

この仮説を「空白仮説」と呼んでいます。AI は、生産性とフローを向上させることで、効率的に働けるようにします。この効率化によって、価値のある仕事を迅速に終えられるようになります。

ここで空白、つまり余剰時間が生まれます。AI は回答者の仕事から価値を奪うのではなく、価値の実現を早めるのです。

ところで、価値のある作業とは？

このように直感に反する調査結果を理解するために、回答者がどのようなことを価値のある作業だと、あるいは骨の折れる作業だと判断しているのか、詳しく調べました。

通常の考え方、過去のレポート、またインタビューから得られた定性的なデータによると、回答者はコーディングのような開発関連のタスクを価値のある作業だと捉えている一方、会議への出席といった組織における調整に関連するタスクなどを、価値のない、骨の折れる作業だと捉えています。この分類スキームでは、回答者が定義した「骨の折れる」作業よりも「価値のある」作業のほうが、AI が支援するのに向いています。

インタビューから得られた定性的なデータに目を向けたところ、自分の仕事を「有意義なもの」と考えるかどうかというモデレーターの質問に回答する際、参加者は自分の仕事の価値を、他人に与える影響と関連させて測っていることがわかりました。

これは、ユーザー中心のアプローチが仕事の満足度に極めて有益な影響を与えるという、DORA が過去 2 年にわたって提示してきたデータによって裏付けられています。

たとえば、P10 は最近の職務内容の変更について説明する際[12]、「そうしたほうが多くの人、多くのことに影響を与えられる」からそのように決断したのだと述べました。同様に P11 は、「一から何かを作り、それが消費者や顧客に届けられるのを見ると、達成感が得られます。自分の中で『よし、自分がこれを提供して、皆がこれを使うんだ』と言えます」と述べました。

開発業務の「意義」は、コードを記述することで直接得られるのではなく、作成したソリューションが与える影響から得られるのだと考えれば、回答者が価値のある作業にかける時間が減る一方、仕事の満足度が高まっている理由を説明できます。

AI は、価値があると考えられているタスクを簡単かつ迅速に行えるようにしますが、そうでないタスクではあまり役立ちません。AI を導入しても骨の折れる作業や燃え尽き症候群が相変わらず存在するということは、退屈な会議や煩雑な作業、その他多くの骨の折れるタスクの解決策を、AI はまだ見出すことができていないことを示しています（図 8）。

幸い、AI は状況を悪化させておらず、回答者のウェルビーイングにも悪影響を与えてはいません。

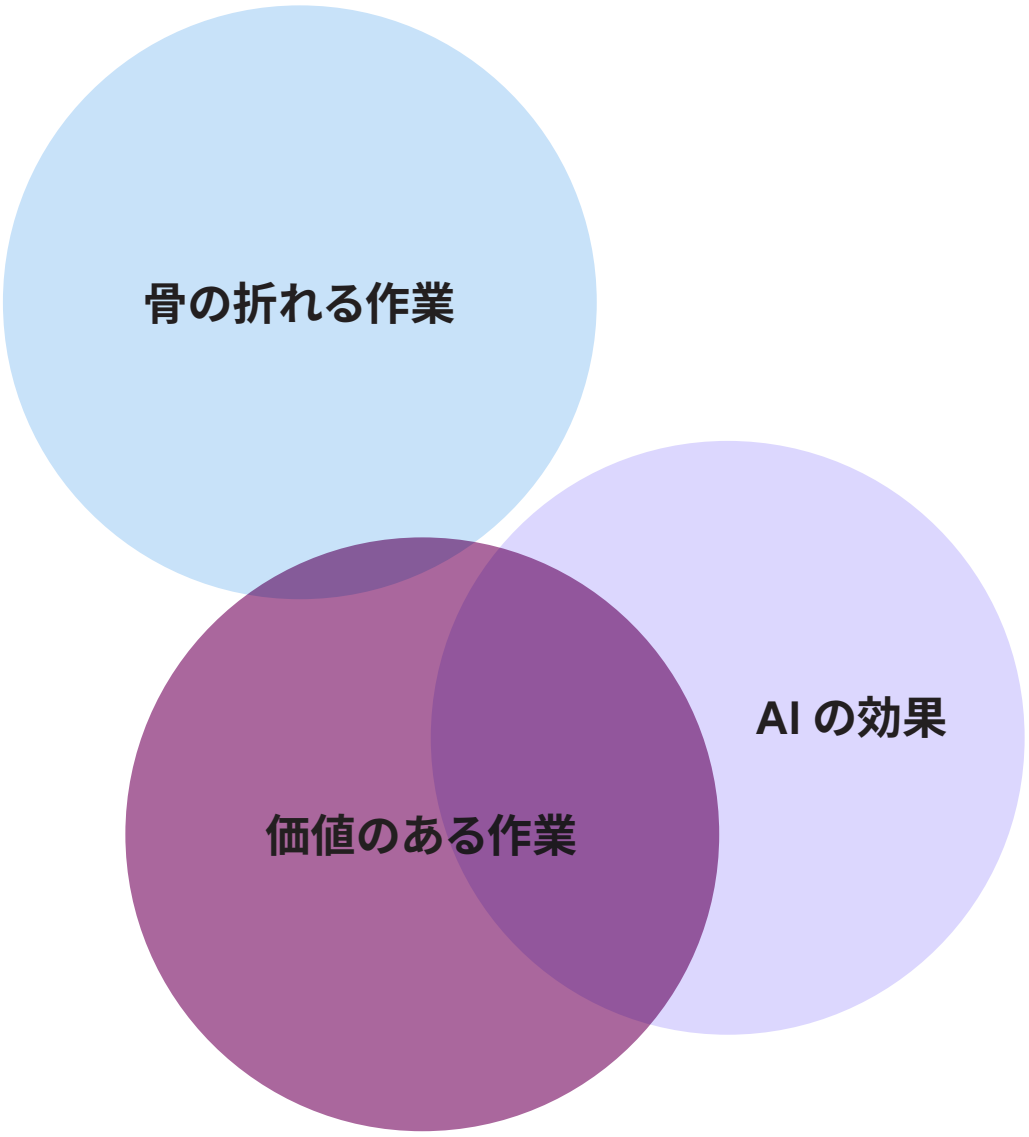


図 8: データではなく仮説を可視化した図。AI は価値のある作業に役立つものであり、骨の折れる作業に役立つものではない。

開発ワークフローに AI がもたらすと期待される影響

前の節では、個人に着目して成果を探りました。今度は、プロセス、コードベース、チームの連携に注目して成果を見ていきます。測定した成果は以下のとおりです。

コードの複雑さ	コードの込み入った仕組みと高度化が生産性の妨げになっている度合い。
既存の技術的問題	過去 6 か月間に、主なアプリケーションまたはサービスに残された技術的問題が生産性を妨げた程度。
コードレビューのスピード	主なアプリケーションまたはサービスのコードレビューを完了するまでに必要な時間の平均値。
承認のスピード	主なアプリケーションまたはサービスについて、コードの変更を提案してから、本番環境での使用が承認されるまでの一般的な期間。
部門横断型チーム（XFN）の協調	「過去 3 か月間、部門横断型チームのメンバーと効果的に連携できた」という文にどの程度同意するか。
コードの品質	過去 6 か月間に、主なサービスまたはアプリケーションの基礎となるコードの品質に満足した、または満足しなかった度合い。
ドキュメントの品質	信頼性、見つけやすさ、更新状況、サポートの提供機能という観点から見た、内部ドキュメント（マニュアル、README、コードのコメント）に対する認識。

目標はこれまでと同様に、AI を導入することで、これらの面に変化が起きるかどうかを把握することです。図 9 に、AI の導入が 25% 増えた場合における、これらの成果の変化に関する最善の推定値を示します。

全体的に、このパターンは AI の非常に説得力のあるストーリーを示唆しています。この節の主な結果は以下のとおりです。

AI の導入が 25% 増えると、関連して以下のようになります。

ドキュメントの品質が 7.5% 向上

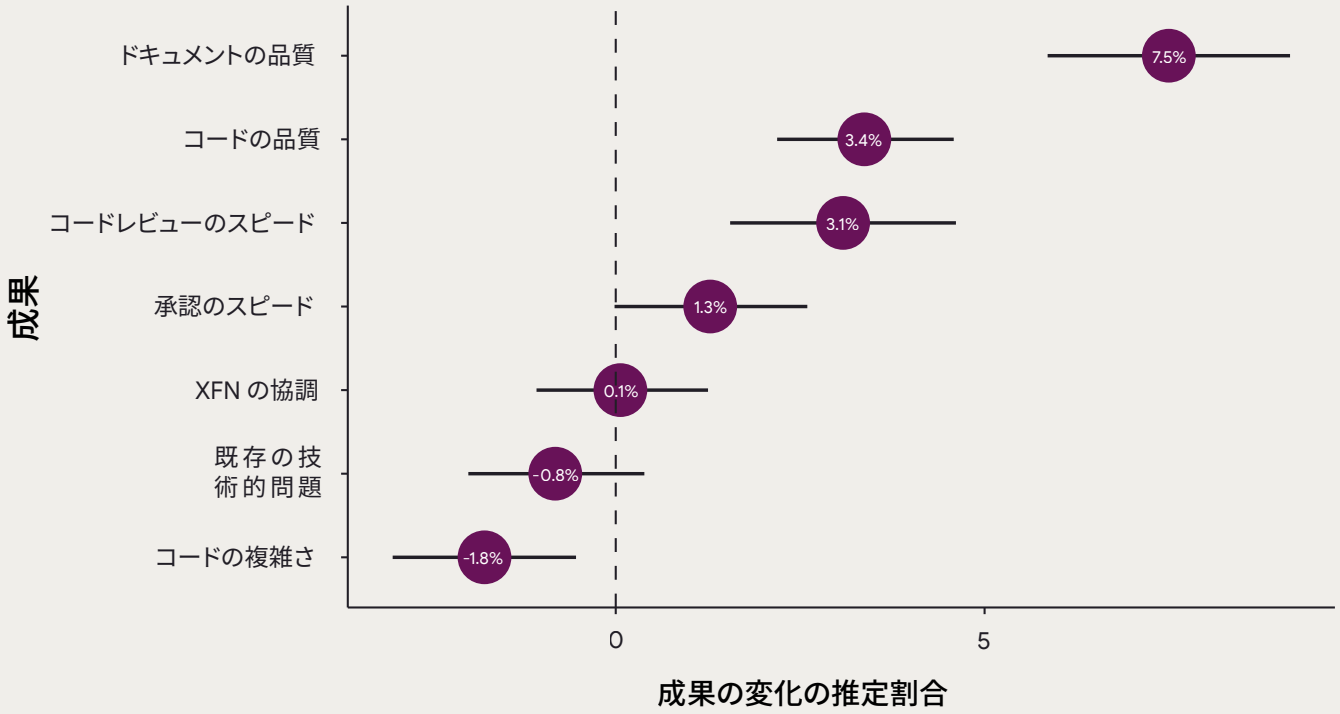
コードの品質が 3.4% 向上

コードレビューのスピードが 3.1% 向上

承認のスピードが 1.3% 向上

コードの複雑さが 1.8% 減少

AI の導入が 25% 増えた場合



点は推定値
エラーバーは 89% 不確実性区間
図 9: AI の導入が組織に与える影響

「AI: 導入と考え方」の章で示したデータによると、AI の最も一般的な用途はコードの記述です。回答者の 67% は、AI がコードの改善に役立っていると回答しています。この感想にはさらなる裏付けがあります。AI でコードの品質が向上し、コードの複雑さが減少するようです（図 9）。古いコードのリファクタリングの見込みと組み合わせると、質の高い AI 生成のコードは、全体的なコードベースの改善につながる可能性があります。そうしたコードベースは、AI を使用して生成された質の高いドキュメントにアクセスしやすくなることで、さらに改善される可能性があります（AI: 導入と考え方参照）。

コードが優れていると、レビューや承認もしやすくなります。AI を活用したコードレビューと組み合わせることで、レビューと承認を迅速に行うことができます。パターンはデータにはっきりと表れています（図 9）。

もちろん、コードレビューや承認が早くなったからといって、コードレビュー プロセスや承認プロセスの質が向上し、さらに徹底されるというわけではありません。プロセスを支援する AI に過度に依存したり、AI 生成のコードを過信したりすることで、スピードが増している可能性もあります。この調査結果は図 9 のパターンと矛盾はしませんが、明白な結論というわけでもありません。

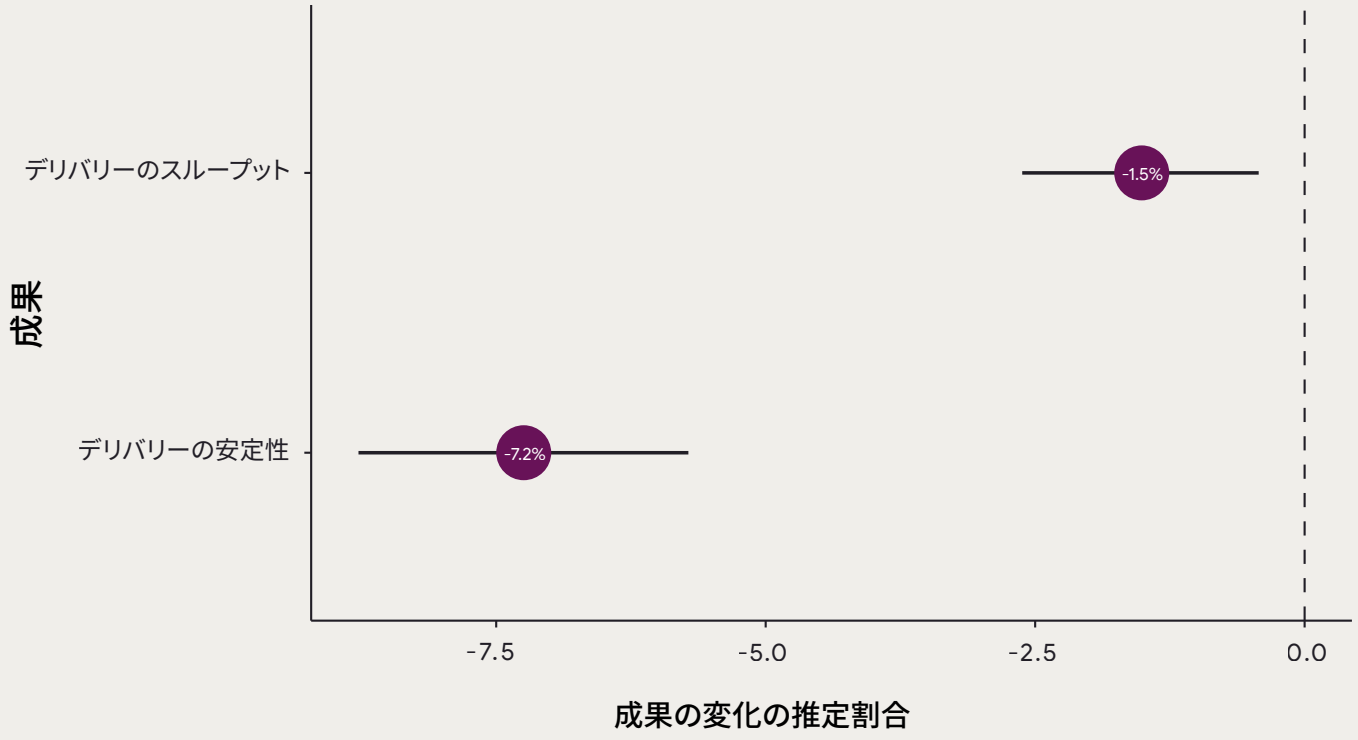
さらに、コードの品質やドキュメントの品質が向上していることが、AI が生成したためなのか、それとも、本来であれば低品質だと見なされていたコードやドキュメントから価値を引き出す能力が AI によって強化されたためなのかは、明らかではありません。AI がコードやドキュメントの理解を助けてくれるがゆえに、AI を利用することで質の高いコードやドキュメントだと見なすしきい値が少し下がるだけだとしたら、どうでしょうか。パターンを理解するためのこれらの 2 つの方法は、互いに排他的な解釈というわけではなく、どちらもパターンに寄与している可能性があります。

これらのパターンから明らかなことは、AI を利用すると、ユーザーが頼りにしているドキュメントや作業対象のコードベースを最大限に活用できるということです。AI は、コードレビューと承認のプロセスから、費用のかさむボトルネックを減らすためにも役立ちます。AI がこれを具体的にどのようにして行っているのか、また、これらの利点がソフトウェア デリバリーの改善などさらなるダウンストリームの利点につながるのかは、明らかではありません。

AI はデリバリー パフォーマンスを低下させている

ここ数年、ソフトウェア デリバリーのスループットとソフトウェア デリバリーの安定性の指標は、互いに独立性を示すようになってきています。スループットと安定性の間に見られた従来の関係性は維持されていますが、新たな調査結果から、これらの因子は十分な独立性を持って機能しているため、別々に考慮する必要があるということが示されています。

AI の導入が 25% 増えた場合



点は推定値

エラーバーは 89% 不確実性区間

図 10: AI の導入がデリバリーのスループットと安定性に与える影響

予想に反して、調査結果から、AI の導入がソフトウェア デリバリー パフォーマンスに悪影響を与えていることがわかりました。デリバリーのスループットに対する影響は小さいものの、おそらくマイナスです（AI の導入が 25% 増えるごとに 1.5% 減ると推定）。デリバリーの安定性には大きな悪影響が及びます（AI の導入が 25% 増えるごとに 7.2% 減ると推定）。このデータを図 10 に示します。

これまでの調査で、ドキュメントの品質、コードの品質、コードレビューのスピード、承認のスピードの改善や、コードの複雑さの減少など、ソフトウェア開発プロセスの改善がソフトウェア デリバリーの改善につながることがわかっています。そのため、AI がこうしたプロセスの尺度を改善する一方、デリバリーのスループットや安定性といったパフォーマンスの尺度を悪化させているようであることは意外でした。

我々は過去の調査結果をもとに、回答者の生産性とコード生成速度に関して AI が根本的なパラダイム シフトを起こしたために、この分野において DORA の特に基本的な原則である「小さいバッチサイズの重要性」が忘れられているのではないか、という仮説を立てました。つまり、AI によって回答者は同じ時間でずっと多くのコードを作成できるようになったため、変更リストのサイズも大きくなっている可能性が高いと言えます。DORA では、大きな変更は遅くなり、また不安定になりやすいことを一貫して示してきました。

まとめると、このデータは、開発プロセスを改善しても、ソフトウェア デリバリーが自動的に改善されるわけではないことを示唆しています。少なくとも、小さいバッチサイズや堅牢なテスト メカニズムなど、ソフトウェア デリバリーを成功させるための基本を確実に守らない限り、改善されることはありません。

AI は、高いソフトウェア デリバリー パフォーマンスを実現するための土台となる、多くの重要な個人的および組織的要因にプラスの影響を与え、この点は楽観的に捉えることができます。しかし、AI は万能薬というわけではないようです。

パフォーマンスの高いチームや組織が AI を利用しているが、プロダクトに恩恵はない？

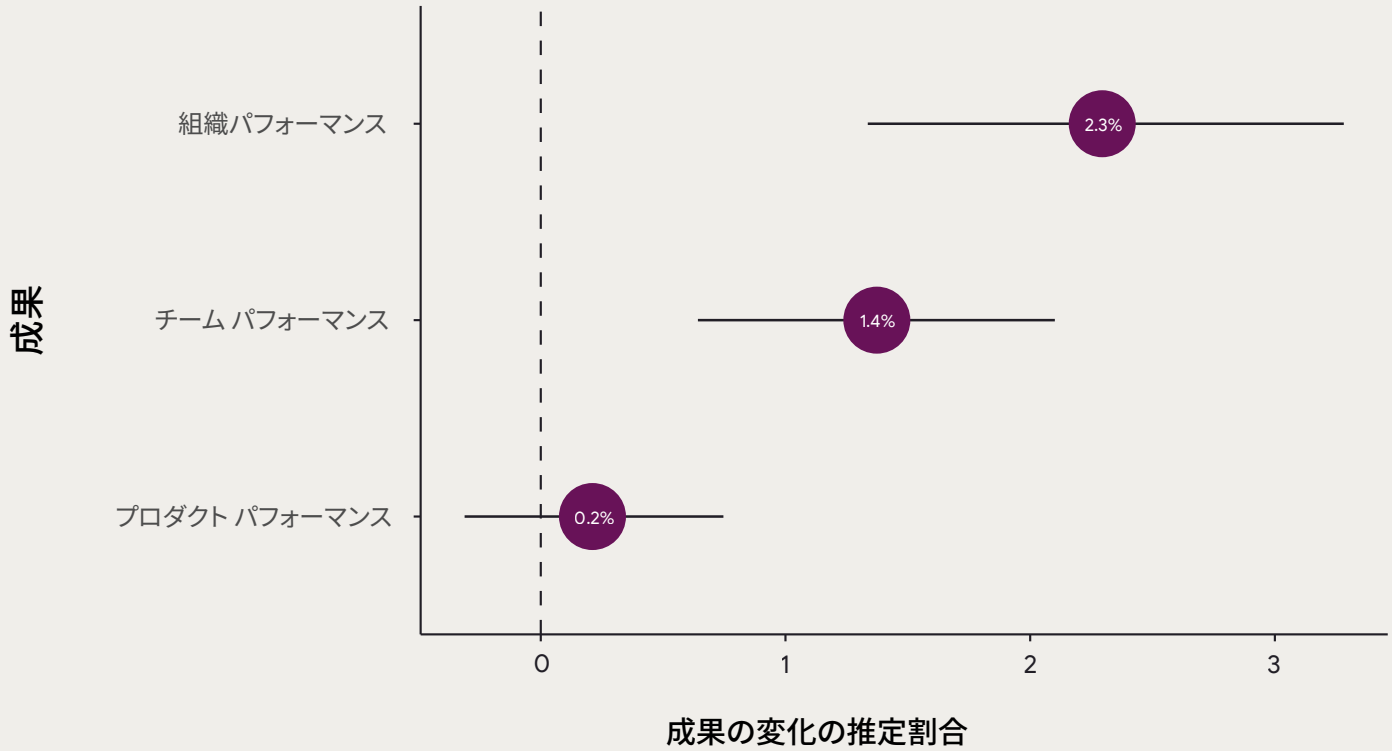
最もダウンストリームの成果と AI の関係

組織パフォーマンス	組織の全体的なパフォーマンス、収益性、マーケット シェア、顧客総数、業務効率、顧客満足度、プロダクトとサービスの品質、目標達成能力を考慮した因子スコア。
チーム パフォーマンス	チームの協調性、革新性、業務効率性、相互信頼性、適応性を考慮した因子スコア。
プロダクト パフォーマンス	プロダクトのユーザビリティ、機能性、価値、可用性、パフォーマンス（レイテンシなど）、セキュリティを考慮した因子スコア。

これらの成果から、AI を導入する個人との関係を導き出すのは難しく、ノイズが多くなります。今日の昼食が今年の組織のパフォーマンスに与える影響を分析しようとしているように感じることがあります。

ミクロレベル（個人など）からマクロレベル（組織など）への飛躍には正当な理由があります。こうした推論上の飛躍については、手法の章をご覧ください。ここでは、関連性を確認してみましょう。

AI の導入が 25% 増えた場合



点は推定値
エラーバーは 89% 不確実性区間
図 11: AI の導入が組織、チーム、プロダクトのパフォーマンスに与える影響

組織レベルのパフォーマンス（AI の導入が 25% 増えるごとに 2.3% 増えると推定）とチームレベルのパフォーマンス（AI の導入が 25% 増えるごとに 1.4% 増えると推定）には、AI の導入の恩恵があるようです（図 11）。しかし、プロダクト パフォーマンスは AI の導入と明らかな関係性はないようです。次に、これらの効果の基となっているものについて理解を試みましょう。

チームや組織のパフォーマンスに寄与する要因は、プロダクト パフォーマンスに影響する要因とは異なる、という仮説を立てました。

チームや組織は、コミュニケーション、知識の共有、意思決定、健全な文化に大きく依存しています。AI はそうした分野のボトルネックを軽減し、チームや組織に良い影響を与える可能性があります。

しかし、プロダクトの成功には追加の要因が関係している可能性があります。優れたプロダクトには、パフォーマンスの高い優秀なチームや組織と同様の根本要因があることは確かですが、開発ワークフローやソフトウェア デリバリーとの間にさらに密接で直接的なつながりがありそうです。そして、どちらも、AI の導入後に安定化している途中かもしれません。

その一部は、優れたプロダクトの基礎をなす技術面の特有の重要性から説明できるかもしれませんが、特に優れたプロダクトの基礎をなすものには、芸術的な側面や共感もあります。すべてが計算で解決できる問題だと考える人には信じがたいかもしれませんが、創造性やユーザー エクスペリエンス デザインなど、プロダクト開発の特定の要素は、今でも（または永遠に）人間の直感や専門知識に大きく依存します。

組織、チーム、プロダクトのパフォーマンスが確かに相互に関係しているという事実に変わりはありません。二変量相関（ピアソン）を見ると、プロダクト パフォーマンスは、チーム パフォーマンス（r=0.56、95% 信頼区間=0.51〜0.60）と組織パフォーマンス（r=0.47、95% 信頼区間=0.41〜0.53）の両方と、中程度の正の相関があることがわかります。

これらの成果は相互に影響し、明確な相互依存関係を生み出します。パフォーマンスの高いチームは優れたプロダクトを開発する傾向がありますが、劣ったプロダクトを受け継ぐと成功が妨げられることもあります。同様に、パフォーマンスの高い組織はリソースとプロセスによってパフォーマンスの高いチームを育成しますが、組織の争いがチームのパフォーマンスを阻害する可能性があります。そのため、AI の導入がチームや組織にとって非常に有益な場合、プロダクトにとっても有益な点が生まれるはずだと考えるのは妥当です。

AI の導入は始まったばかりです。AI の影響に固有の性質により、または効果的な活用に関連した学習曲線が原因で、一部のメリットやデメリットが表れるまで時間がかかるかもしれません。

プロダクトのイノベーションや開発における AI の潜在能力を完全に実現する前に、AI が組織やチームにどのように役立つかを知ろうとしている、というだけのことかもしれません。これがどのように展開していくのかを示そうとしたのが図 12 です。

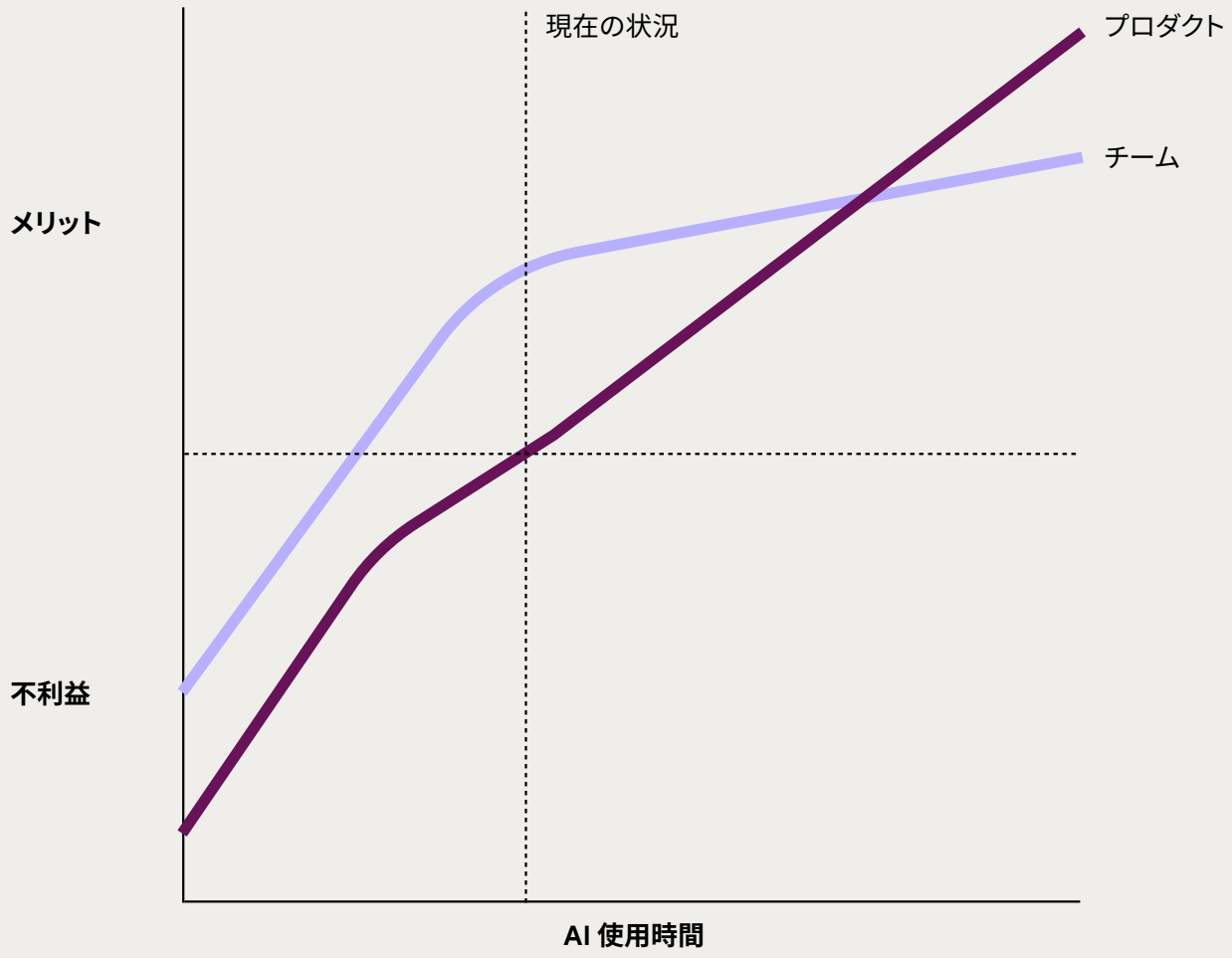


図 12: さまざまな学習曲線の表現。説明のための抽象的な表現です。実際のデータに基づくものではありません。

次のステップ

DORA では、AI が個人、チーム、組織を現在どのように支援できるのか、その可能性を理解したいと考えました。表れたパターンは、AI の可能性はただのでたらめではないことを裏付けています。実際に何かが起きているのです。

AI の導入を支持する明確な兆候があります。とはいえ、障害となり得るもの、成長に伴う課題、AI が悪影響を及ぼす可能性がたくさんあることも、非常に明らかです。

AI を大規模に導入することは、再生ボタンを押すような簡単なことではないかもしれません。注意深く計画された臨機応変で透明性のある戦略は、大きな利益をもたらす可能性を秘めています。この戦略は、リーダー、チーム、組織、研究者、AI 開発者によって共同で策定する必要があります。

リーダーや組織は、従業員に最もメリットがある分野での導入を優先する方法を見つける必要があります。

AI 導入戦略の方向性を決める方法のアイデア

AI に関する明確な使命とポリシーを定めて組織とチームを強化する

AI に関する使命、目標、導入計画について、透明性の高い情報を従業員に提供します。包括的なビジョンと具体的なポリシーの両方を明確にすることで、許可されたコードの配置や利用可能なツールといった手続き上の懸念事項に対処して、不安を軽減するとともに、価値のある充実したクリエイティブな作業に誰もが集中できるようにする手段として AI を位置づけることができます。

AI に関する継続的な学習と試験運用の文化を築く

個人やチームが有益なユースケースを見つけるための時間を確保し、それぞれに道筋をつける自主性を持たせることで、AI ツールの継続的な探求を促す環境を醸成します。サンドボックス環境や低リスク環境での実践経験を通じて、AI テクノロジーに対する信頼を築きます。自動化された堅牢なテストの開発に注力し、リスクをさらに軽減することを検討します。導入実績だけで評価するのではなく、ダウンストリームに及ぼす有意義な影響（従業員が活躍できる、プロダクトの利用者に有益、チームの潜在能力を引き出す、など）で AI を評価する測定フレームワークを実装します。

AI のトレードオフを認識して活用することで競争優位性を確保する

潜在的な欠点（価値のある作業にかける時間が減る、AI への過度の依存、ある分野で得た利益が他では課題になる可能性、ソフトウェア デリバリーの安定性とスループットへの影響）を認識することで、陥りやすい問題を回避し、組織やチームで AI に前向きな道筋をつける機会を特定できます。AI がいかに有益になりうるかだけでなく、いかに有害になりうるかについても理解を深めることで、習熟を早め、探求をサポートして、学習したことを行動や真の競争優位性につなげることができます。

期待できることも学ぶべきことも、たくさんあることは明らかです。DORA はこの 10 年間と同様に、今後も率直かつ正確で有益な視点を提供することに注力し、最善を尽くしていきます。

1. https://www.goldmansachs.com/insights/top-of-mind/gen-ai-too-much-spend-too-little-benefit
2. https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand
3. https://www.washington.edu/news/2023/07/27/how-much-energy-does-chatgpt-use/
4. https://www.gatesnotes.com/The-Age-of-AI-Has-Begun
5. https://www.businessinsider.com/ai-chatgpt-homework-cheating-machine-sam-altman-openai-2024-8
6. https://www.safe.ai/work/statement-on-ai-risk
7. https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/
8. https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality
9. https://www.nytimes.com/2024/04/15/technology/ai-models-measurement.html
10. https://dora.dev/capabilities
11. これは特別なアプローチではありませんが、この分野ではやや珍しいアプローチであることを明確にしておく必要があります。
12. （P1）など、（P[N]）は、インタビュー参加者の仮名を表しています。

プラットフォーム エンジニアリング

はじめに

プラットフォーム エンジニアリングは、業界全体で関心を集め、勢いを増している、新しいエンジニアリング分野です。Spotify や Netflix などの業界のリーダーや Team Topologies などの書籍が刺激になっています[1]。

プラットフォーム エンジニアリングは、さまざまなチーム間の社会的インタラクションと、自動化、セルフサービス、プロセスの再現性といった技術的側面との関わりに着目する社会技術的な分野です。プラットフォーム エンジニアリングの基本概念については、DORA によるものを含め、長年にわたって研究が行われてきました。

DORA の調査では主に、ソフトウェアを外部ユーザーに提供する方法に注目しています。一方、プラットフォーム チームのアウトプットは通常、内部に向けられたものであり、ソフトウェアの開発と運用のライフサイクルをサポートするように設計された API、ツール、サービスなどです。

プラットフォーム エンジニアリングでは、ゴールデンパス（アプリケーションの提供や運用に必要なリソースを扱うときにプラットフォームのユーザーが使用する、高度に自動化されたセルフサービス ワークフロー）を構築してデベロッパー エクスペリエンスを改善することに注力します。目的は、ソフトウェアの構築と提供の複雑性をなくし、開発者がコードだけに集中できるようにすることです。

ゴールデンパスで自動化されるタスクとしては、新しいアプリケーションのプロビジョニング、データベースのプロビジョニング、スキーマ管理、テスト実行、ビルドおよびデプロイ インフラストラクチャのプロビジョニング、DNS 管理などが挙げられます。

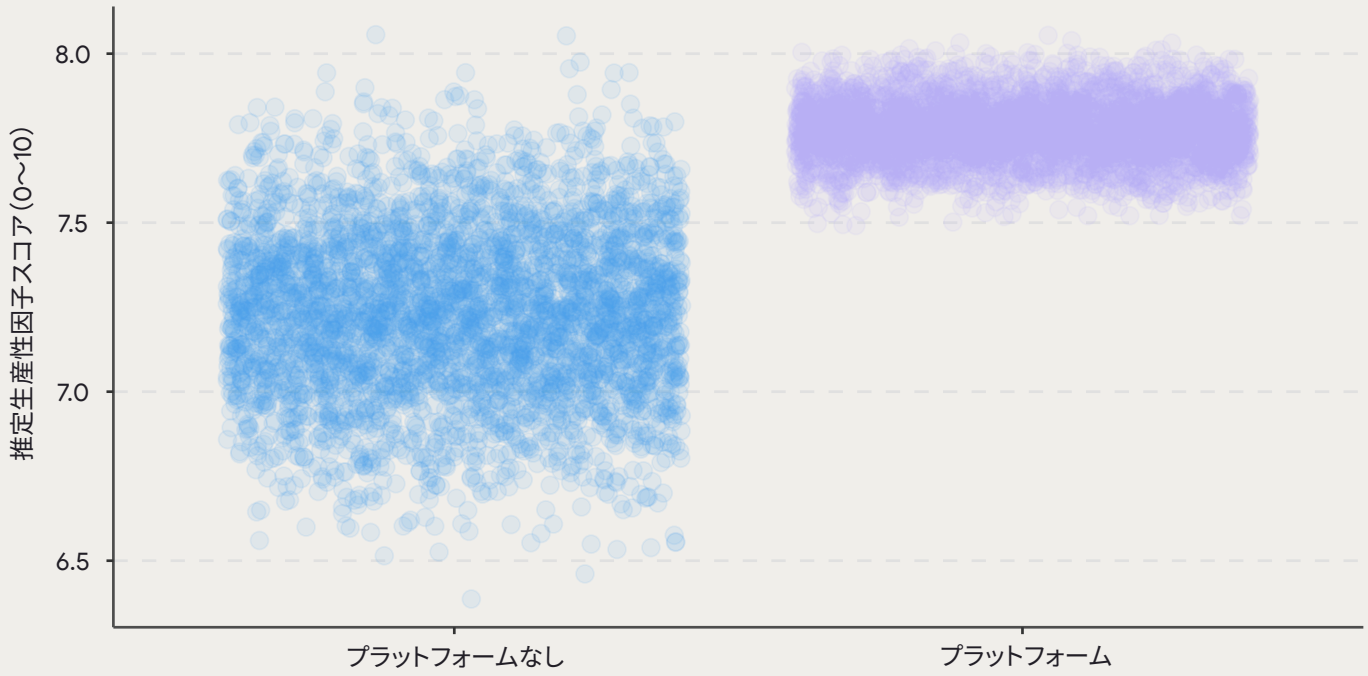
共有システムへの機能の移行（「シフトダウン」ともいいます）などのプラットフォーム エンジニアリングのコンセプトは[2]、「自ら開発し、自ら運用する」といったアプローチに反しているように思われるかもしれません。しかし、DORA ではプラットフォーム エンジニアリングを、組織全体にこうしたアプローチの導入を拡大するための方法だと捉えています。機能が一旦プラットフォームに組み込まれれば、チームはそのプラットフォームを導入することで、機能を基本的に無料で利用できるからです。

たとえば、単体テストを実行して結果を開発チームに直接報告する機能がプラットフォームにあり、テスト実行環境の構築と管理を開発チームが行う必要はない場合、チームは継続的インテグレーション プラットフォームの機能により、質の高いテストを作成することに集中できます。この例では、継続的インテグレーション機能を大規模な組織全体に拡大し、複数のチームが継続的テストやテストの自動化によってチームの能力を簡単に向上させることができます[3,4]。

成功の鍵は、ユーザー中心（社内開発者プラットフォームにおけるユーザーは開発者）、開発者の独立性、プロダクト マインドセットでプラットフォーム エンジニアリングにアプローチすることです。今年も例年もユーザー中心が組織のパフォーマンスを向上させる重要な要素だと認識されていることを考えれば、これは驚くようなことではありません[5]。ユーザー中心のアプローチがなければ、プラットフォームは助けになるというよりも、むしろ邪魔になります。

今年のレポートでは、プラットフォームのソフトウェア デリバリーと運用パフォーマンスとの関係をテストするよう努めました。その中で、良い結果が得られています。社内開発者プラットフォームのユーザーは、個人の生産性が 8% 向上し、チームのパフォーマンスが 10% 向上しました。さらに、プラットフォームを使用することで、組織のソフトウェア デリバリーおよび運用パフォーマンスが 6% 向上しています。しかし、メリットだけで欠点がないわけではありません。意外にも、スループットは 8% 低下し、変更の安定性は 14% 低下しました。

以降の各節では、今回のアンケートで明らかになった数値、微妙な差異、意外なデータについて詳しく説明します。プラットフォーム エンジニアリングの取り組みを始めたばかりの場合でも、長年にわたって行っている場合でも、主な調査結果を応用することで、プラットフォームを改善できます。



各ドットは最も妥当な平均生産性スコア推定値 8,000 個のうちの一つ
図 13: 社内開発者プラットフォームを使用した場合と使用しなかった場合における個人の生産性の因子

プラットフォーム エンジニアリングの確かな可能性

社内開発者プラットフォームは、実践を通じて効率と生産性の向上を果たし得ることから、ソフトウェア デベロッパーや IT 業界の幅広いセクションから関心を集めています。今年のアンケートでは、社内開発者プラットフォームの定義を極めて広いままとしたところ[6]、回答者の 89% が社内開発者プラットフォームを利用していることが判明しました。この母集団の中で、インタラクション モデルは非常に多様です。

プラットフォーム エンジニアリングに対する業界の幅広い関心レベルや、この分野の新規性を反映したデータポイントとなっています。

全体的にプラットフォームの影響はプラスであり、社内開発者プラットフォームを使用した場合、個人の生産性は 8% 向上し、チームのパフォーマンスは 10% 向上しました。

生産性だけでなく、組織全体のパフォーマンスについても、プラットフォームを使用することで 6% の向上が確認されています。全体として、組織はプラットフォームにより、ソフトウェアを迅速に提供し、ユーザーのニーズに応え、ビジネス価値を高めることができます。

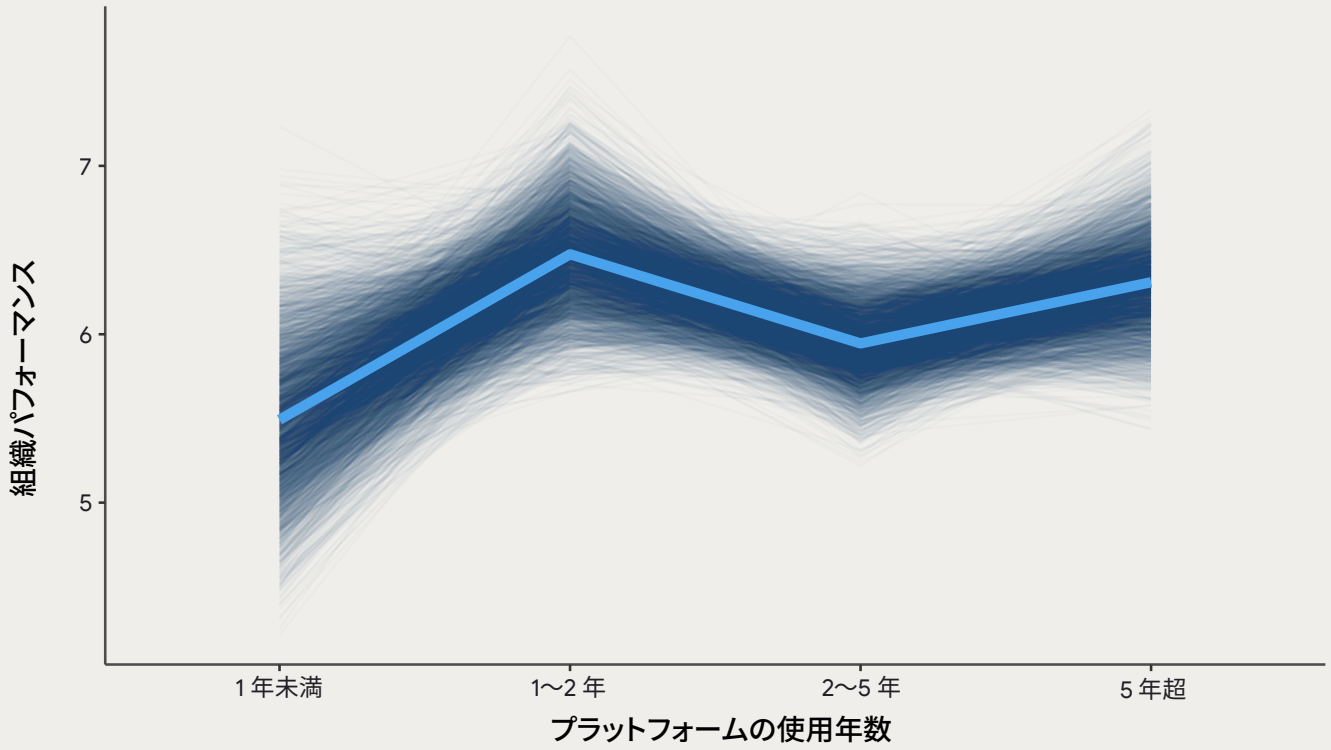
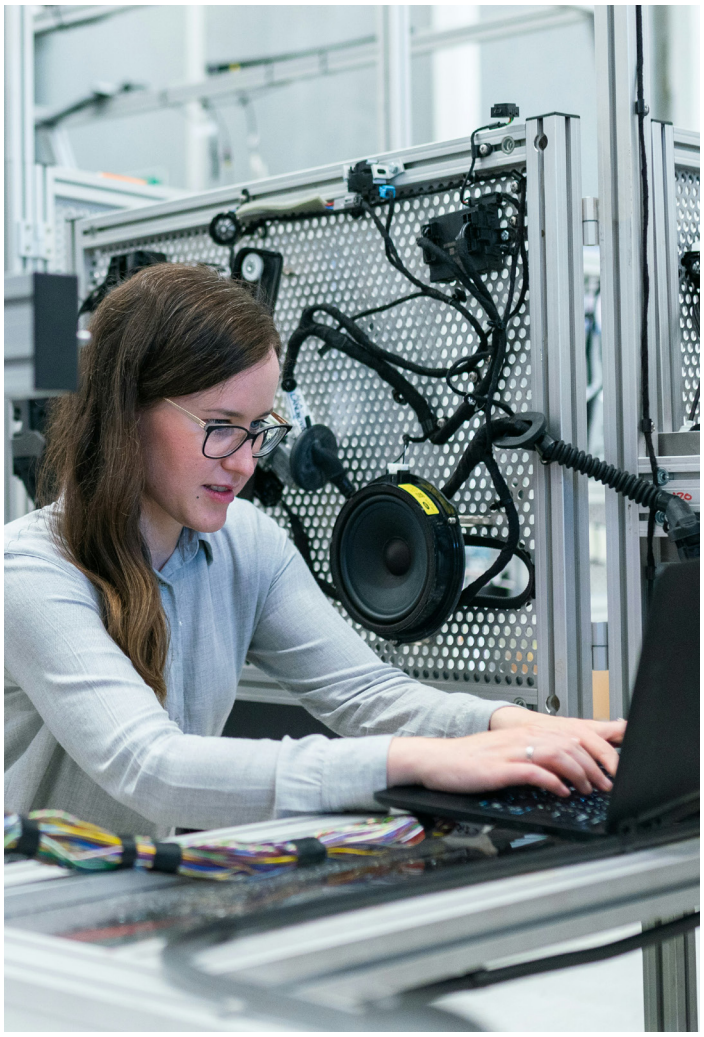


図 14: 社内開発者プラットフォームを使用した場合の、プラットフォームの使用年数による組織パフォーマンスの違い

生産性を見るときにプラットフォームの使用年数を考慮すると、プラットフォーム エンジニアリングの取り組みを開始したときに初期のパフォーマンスの向上が見られ、使用年数が長くなり成熟するのに伴って、いったん低下した後にまた回復しています。これは、初期には有益であるものの、実現した後で課題に直面するという、変革の取り組みでよく見られるパターンです。

長期的に見れば生産性の向上は維持されており、ソフトウェア デリバリーおよび運用プロセスにおける社内開発者プラットフォームの役割について、全体的な可能性が見て取れます。



主な調査結果 - 開発者の独立性の影響

開発者の独立性は、社内開発者プラットフォームを使用してソフトウェアを提供する際に、個人レベルでもチームレベルでも生産性に大きな影響を与えました。開発者の独立性は、「開発者がアプリケーション ライフサイクル全体にわたって、イネーブリング チームに依存することなく、タスクを遂行する能力」と定義されています。

プラットフォームのユーザーがイネーブリング チーム（支援を専門とするチーム）の関与なしでタスクを完了できる場合、チームと個人の両方のレベルで生産性が 5% 向上しています。この調査結果は、セルフサービス ワークフローの実現に注力するという、プラットフォーム エンジニアリングにおける重要な原則を改めて示しています。

プラットフォーム チームにとって、これはプラットフォーム エンジニアリング プロセスの重要な部分、つまりユーザー フィードバックの収集を指しているため重要です。アンケートの回答では、どのような形式のフィードバックが最も効果的かは示されていませんが、一般的な方法としては、非公式の会話と公開バグトラッカー、次いで、継続的な共同開発、アンケート、テレメトリ、インタビューが挙げられます。

こうした方法はいずれも、ユーザーが独立してタスクを完了できているかどうかを把握する際に有効です。アンケートのデータからは、プラットフォームでフィードバックを収集しなかった場合、悪い影響があることもわかりました。

副次的な調査結果 - 専任のプラットフォーム チームの影響

興味深いことに、専任のプラットフォーム チームを設けることによる生産性への影響は、個人の場合ごくわずかでした。その一方で、チームのレベルでは生産性が 6% 向上する結果となっています。専任のプラットフォーム チームを設けることは個人にとって有益なものの、チーム全体ではさらに大きな影響があるということが示されており、影響が一様ではないため、この調査結果は意外でした。

チームにはさまざまな職務とスキルを持つ開発者が複数いるため、個人のエンジニアと比較すると、当然ながらタスクが多様になっています。専任のプラットフォーム エンジニアリング チームを設けると、プラットフォームがチームのタスクの多様性により適切に対応できるようになるという説明が考えられます。

全体としては、社内開発者プラットフォームを用意すれば生産性にプラスの影響があると言えます。

要点:

セルフサービスと自律的に完了できるワークフローで開発者の独立性を実現するユーザー中心のアプローチが重要です。プラットフォームにおいて、ユーザーとは社内のエンジニアリング チームと開発チームのことです。

他の変革と同様に、プラットフォーム エンジニアリングにも「J 曲線」が当てはまるため、生産性の向上は継続的な改善によって安定します。

予期せぬ欠点

プラットフォーム エンジニアリングには、チームや個人の生産性が向上し、組織のパフォーマンスが向上するという明確な利点がある一方、予期せぬ欠点もありました。スループットと変更の安定性が低下することがわかったのです。

予期せず、変更の不安定性と燃え尽き症候群の間に大変興味深い関係性が見つかりました。

スループット

スループットに関しては、プラットフォームを使用しない場合と比較して、約 8% の低下が確認されました。DORA はその原因について、次のような仮説を立てました。

まず、本番環境に変更をデプロイする前に必要なプロセスが増えることで、変更全体のスループットが低下します。一般的に、社内開発者プラットフォームがソフトウェアの構築と提供に使用されている場合、システム間、そして暗黙的にチーム間で、「受け渡し」の数が増加します。

たとえば、コードがソース管理に commit されると、テスト、セキュリティ チェック、デプロイ、モニタリングのために、さまざまなシステムが自動的にコードを取得します。

このような受け渡しのそれぞれが原因となってプロセス全体で時間がかかることになり、スループットが低下しますが、ソフトウェアを完成させる機能そのものは実質的に向上します。

次に、「アプリケーション ライフサイクル全体で、タスクに必ずプラットフォームを使用する」ことが必須となっていると回答した回答者の場合、スループットが 6% 低下していました。決定的な関係ではないものの、これは 1 つ目の仮説と関連している可能性があります。

プラットフォームがあることでソフトウェアの開発とリリースに関係するシステムとツールが増えるのであれば、目的に合わない可能性があってもそのプラットフォームを使用せざるを得ないことや、プロセス遅延の自然な増加によって、プラットフォームの例外のない使用と生産性低下の関係を説明できるかもしれません。

これに対処するには、ユーザー中心のアプローチで、ユーザーの独立性を確保するようにプラットフォーム エンジニアリングに取り組むことが重要です。

変更の不安定性と燃え尽き症候群

社内開発者プラットフォームを使用した場合の、開発時と運用時におけるアプリケーションの変更の安定性を検討したところ、変更の安定性が 14% も低下することがわかりました。これは、プラットフォームを使用した場合、変更時の障害率とやり直し率が大幅に増加するということを示しています。

さらに興味深いことに、不安定性とプラットフォームが組み合わさると、燃え尽き症候群の度合いが増えることがわかりました。プラットフォームが燃え尽き症候群につながるわけではありませんが、燃え尽き症候群に関しては、不安定性とプラットフォームの組み合わせが特に問題となります。スループットの低下と同様に、燃え尽き症候群に変化が起こる理由について完全にはわかっていませんが、仮説を立てました。

1 つ目は、開発者やチームは、プラットフォームがあることで、変更を push する際に問題があってもすぐに修正できるという自信をより強く持つようになるというものです。この場合、プラットフォームがあることでチームが試験的な変更と提供を行えるようになり、その結果として変更時の障害率ややり直し率が高くなっていることになるため、不安定性の度合いが高いことは必ずしも悪いことではありません。

2 つ目の仮説は、変更や本番環境へのデプロイの品質を確保するにあたり、プラットフォームは効果的ではないというものです。

または、プラットフォームが、アプリケーションに含まれるあらゆるテストを実施する自動テスト機能を提供していて、それにもかかわらず、アプリケーション チームが質よりもスループットを優先したり、テストの改善を行わなかったりと、その機能を十分に活用していない可能性もあります。どちらのシナリオでも、不適切な変更が実際にプロセスを通過し、やり直しにつながります。

また、3 つ目の仮説として、変更の不安定性と燃え尽き症候群の度合いが高いチームは、安定性を向上させ燃え尽き症候群を減らすことを目的としてプラットフォームを作成する傾向がある、ということも考えられます。プラットフォーム エンジニアリングは、燃え尽き症候群を減らし、小さな変更を一貫して提供する能力を高める手法と見なされることが多いため、これは理にかなっています。この仮説に基づくと、プラットフォーム エンジニアリングは、燃え尽き症候群や変更の不安定性を抱えた組織の兆候ということになります。

初めの 2 つのシナリオでは、プラットフォームで許容されるやり直しが負担となり、燃え尽き症候群が増えている可能性もあります。特に、プラットフォームが不適切な変更を可能にしている 2 つ目のシナリオのほうが燃え尽き症候群につながりやすそうですが、どちらのシナリオでも、チームや個人は変更や機能を push できるため、生産的だと感じられることはありうるでしょう。3 つ目のシナリオでは、変更の不安定性と燃え尽き症候群はプラットフォーム エンジニアリングの取り組みの予測因子であり、プラットフォームはこうした変更の解決策と見なされます。

トレードオフのバランス

プラットフォーム エンジニアリングは万能薬ではありませんが、ソフトウェアの開発と運用のプロセス全体に関して言えば、有望な分野となる可能性があります。他の分野と同様に、プラットフォーム エンジニアリングにも利点と欠点があります。

DORA の調査によると、プラットフォーム エンジニアリングの取り組みを始めるとき、トレードオフのバランスをとるためにできることは 2 つあります。それらを実行することで、組織はプラットフォーム エンジニアリングの利点を活かしつつ、潜在的な欠点をモニタリングし、管理することができます。

まず、開発者の独立性とセルフサービス機能を実現するプラットフォーム機能を優先します。その際、プラットフォームをアプリケーション ライフサイクルのあらゆる面に使用するよう厳密に要求すると、開発者の独立性が阻害されるおそれがあるため、バランスに注意してください。

ユーザーがプラットフォームで提供されるもの以外のツールや自動化を利用できるよう、選択肢を用意しておくとよいでしょう。ただし、独立性が向上する代わりに複雑性が増します。このトレードオフは、プラットフォームのユーザーと積極的に協力してフィードバックを収集する専任のプラットフォーム チームによって軽減できます。

協力とフィードバックは、プラットフォームの取り組みのユーザー中心度を高め、プラットフォームの長期的な成功に寄与します。データからわかるように、フィードバックにはさまざまな収集方法があります。フィードバックをできるだけ多く収集するために、複数の方法を採用してください。

次に、アプリケーションの変更の不安定性を注意深くモニタリングし、発生した不安定性が意図的なものかどうかを把握するよう努めます。プラットフォームは、不安定性と引き換えに試験的な運用を促し、生産性を向上させ、パフォーマンスを大幅に高める可能性があります。

一方、この不安定性によって燃え尽き症候群が発生する可能性もあるため、プラットフォーム エンジニアリングの取り組みの全体を通じて注意深くモニタリングし、考慮する必要があります。その場合、不安定性の許容範囲を把握しておくことが重要です。サービス レベル目標（SLO）と、サイト信頼性エンジニアリング（SRE）のエラー バジェットを使用すると、試験的な運用を安全に行うにあたり、リスク許容度とプラットフォームの有効性を測ることができます。

社内開発者プラットフォームは開発者のエクスペリエンスに重きを置いていますが、ソフトウェアの提供と運用を効果的に行うには、他にも多くのチームが必要となります（データベース管理者、セキュリティ、運用など）。

プラットフォーム エンジニアリングの取り組みでは、全チームを横断し、組織の目標に沿って、ユーザー中心の文化と継続的な改善を促進します。

そうすることで、プラットフォームの機能、サービス、API を調整し、ソフトウェアとビジネス価値の提供に取り組む個人やチームのニーズに適切に対応することができます。

1. Skelton, Matthew and Pais, Manuel.2019.Team Topologies: Organizing Business and Technology Teams for Fast Flow.IT Revolution Press. https://teamtopologies.com/
2. https://cloud.google.com/blog/products/application-development/richard-seroter-on-shifting-down-vs-shifting-left
3. https://dora.dev/capabilities/continuous-integration/
4. https://dora.dev/capabilities/test-automation/
5. https://dora.dev/research/2023/https://dora.dev/research/2016/
6. https://dora.dev/research/2024/questions/#platform-engineering

開発者エクスペリエンス

要点

ソフトウェアは勝手にできあがるものではありません。たとえ AI の支援を利用したとしても、ソフトウェアを構築するのは人であり、彼らの業務におけるエクスペリエンスは組織が成功するための基礎的な要素です。

今年のレポートでも、開発者が構築するものとユーザーが必要とするものが一致すれば、従業員も組織も成功するという結果が得られました。開発者は、ユーザー中心の考え方に基づいてソフトウェアを構築すれば、生産性が向上し、燃え尽き症候群に陥りにくくなり、質の高いプロダクトを構築しやすくなります。

ソフトウェアはユーザーのために構築されるものであるため、開発者がユーザー エクスペリエンスに優れたソフトウェアの構築に注力できるような環境を醸成することは組織の責任です。また、優先順位が常に変動しているような状況のない安定した環境は、わずかながら生産性の有意な向上につながり、従業員の燃え尽き症候群の重大かつ有意な減少につながることがわかっています。

環境要素は、開発するプロダクトの品質や、プロダクトの構築に従事している開発者の全体的なエクスペリエンスに大きな影響を与えます。

ユーザーを第一に考えることで他の（ほとんど）すべてがうまくいく

開発者の仕事は、素晴らしいものです。開発者は技術的進歩の最前線に立ち、私たちの生き方、働き方、世界との関わり方を形作ることに貢献しています。

開発者の仕事は、基本的には人、つまり作成するソフトウェアやアプリケーションのユーザーと結びついています。しかし開発者は多くの場合、機能やイノベーションを優先する環境で働いています。作成したプロダクトを使用する人にそうした機能が価値をもたらすかどうかを把握することは、それに比べると重視されていません。

ここでは、エンドユーザーを優先するソフトウェア開発のアプローチが従業員にも組織にもプラスの影響を与えることを示す、説得力のある根拠を提示します。

今年は、開発者について以下のことが言えるかどうかを把握するための質問を行いました。

1. ユーザーからのフィードバックを取り入れて、機能の再考や優先順位の見直しを行っている

2. ユーザーが特定のアプリケーションやサービスで何を達成しようとしているのかを把握している

3. ユーザーを重視することがビジネスの成功の鍵だと考えている

4. ユーザー エクスペリエンスはビジネスの最優先事項だと考えている

調査結果とその意味

データは、ユーザーのニーズや課題を指針として活用している組織がより優れたプロダクトを生み出すということを強く示唆しています。

ユーザーを重視すると生産性と仕事の満足度が向上し、燃え尽き症候群のリスクが減少することがわかりました。

重要なのは、こうした利点は個々の従業員だけでなく、組織にも及ぶということです。例年、パフォーマンスの高い組織はソフトウェアを迅速かつ確実に提供しているということを強調してきました。つまり、ソフトウェア デリバリー パフォーマンスが成功のための要件だということです。

しかし、データは成功につながる別の道があることも示しています。

開発者、その雇用主、組織全般は、ユーザー中心のソフトウェア開発アプローチを築くことができます。

ユーザーのニーズを知って理解している組織では、ソフトウェア デリバリーの安定性とスループットはプロダクトの品質の要件ではないということがわかりました。ユーザー エクスペリエンスが優先されている限り、プロダクトの品質は高くなります。

組織がユーザーを重視せず、ユーザーのフィードバックを開発プロセスに取り入れていない場合、安定性と迅速な提供を強化することが、プロダクトの品質を確保する唯一の方法となります（図 15 参照）。

組織によっては機能の開発や技術革新に注力する必要があるという傾向は理解できます。額面通りに考えれば、このアプローチは理にかなっています。結局のところ、開発者が平均的なユーザーよりはるかにテクノロジーに精通していることは間違いありません。

しかし、ユーザー エクスペリエンスに関する仮定や思い込みだけに基づいてソフトウェアを開発すると、開発者は、良さそうに見えてもほとんど使われない機能を作ってしまう可能性が高くなります[1]。

組織や従業員がユーザーの実世界での体験を理解すれば、ユーザーの真のニーズに対応した機能を構築できる可能性が高まります。ユーザーの真のニーズに対応すれば、その機能が実際に使われる可能性が高まります。

ユーザーのために構築することを重視すれば、快適なプロダクトが生まれます。

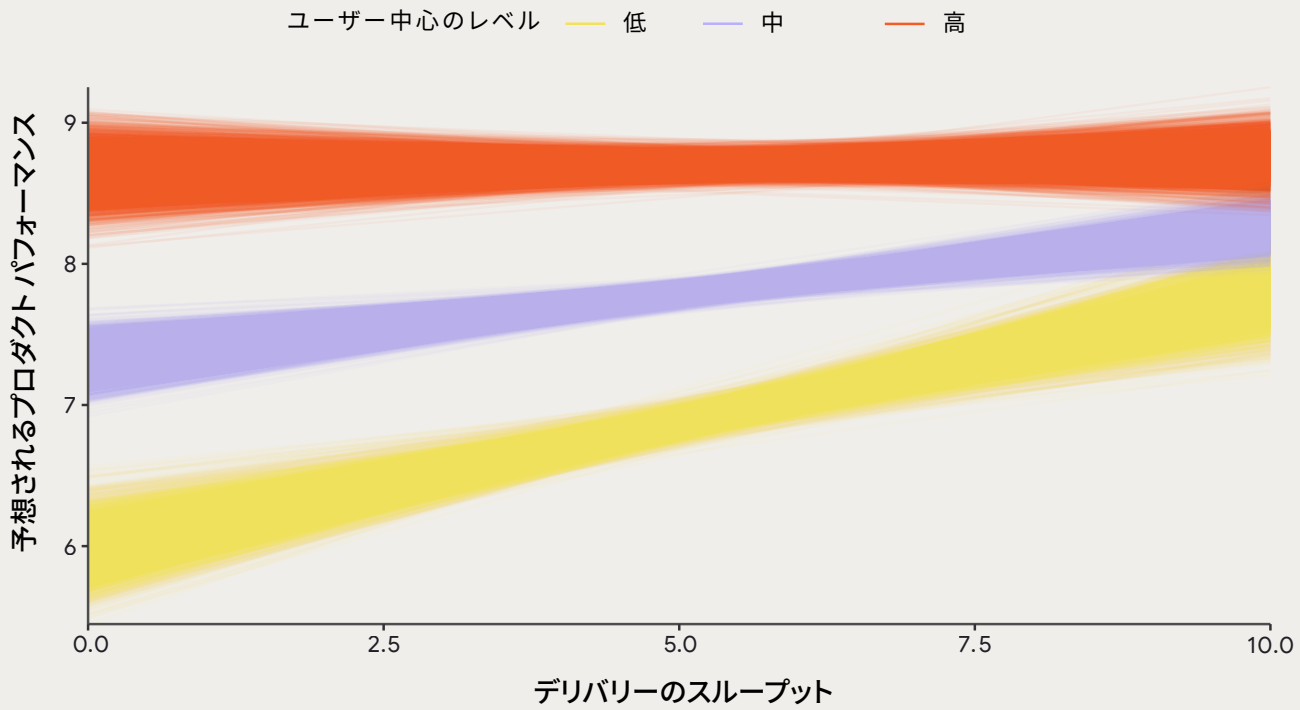


図 15: 3 つのレベルのユーザー中心アプローチにおけるプロダクト パフォーマンスとデリバリー スループット

ユーザー中心のソフトウェア開発アプローチが哲学や手法としてこれほどまでに強力である理由

学術研究によれば、仕事から目的意識を引き出すことは従業員や組織にとって有益です[2,3]。

たとえば最近の調査では、労働者の 93% が、自分の業務に意味があると感じられる仕事をすることが重要であると回答しています[4]。同様に別の調査では、回答者は平均して、常に有意義な仕事ができるのであれば将来の全収入の 23% を放棄しても構わないと回答しています[5]。

この従業員の意向は、驚くべきトレードオフです。人を動機付ける要因の一端がこれでわかります。また、人は有意義なことに時間を使いたいと考えていることが伺えます。

「誰もが社外の個人や地域社会にプラスの影響を与えるような会社で働ければよいのですが、それが実現するとは限りません。そのようなことが常に可能なわけではありません。自動運転に関する壮大なビジョンの多くは、運転できる人が、高速道路を走りながら眠れるようになるというものです。でも、私はそんなことのために働いているわけではありません。運転できない人が、好きな場所へ行って、好きなことを自由にできるようにしたいのです。」(P2)[6]

方向性を明確に示す

ユーザー中心のソフトウェア開発アプローチは、開発者の仕事に対する見方を根本的に変えることができます。独断的な機能をリリースし、ユーザーがそれを使うかどうかを推測するのではなく、ユーザーのフィードバックに基づいて、作成するものに優先順位を付けることができます。

このアプローチで開発者は、取り組んでいる機能には存在意義があるのだという確信を得られます。すると突然、仕事に意味が生まれます。ユーザーがプロダクトやサービスを使用する際に最高の体験ができるようにする、ということです。もはや、開発されたソフトウェアとそれが存在する世界の間に隔たりはありません。

開発者は作成したソフトウェアを通じて、自分たちの仕事の直接的な影響を知ることができます。

「私たちは、企業として、成果を出すようプレッシャーにさらされています。そのため、見かけ上は優れているようなプロダクトも、どのように改善したいかという論点においても、最近の構造変化によって、品質ではなく提供が重視されるようになりました。私としては、これは大きな悩みの種です。」(P9)

部門横断的な連携を促進する

どれほど優秀な開発者でも、一人でソフトウェアを構築することはありません。質の高いプロダクトを構築するには、能力がさまざまでも互いに補い合えるような、たくさんの人たちの協力が必要です。

ユーザー中心の開発アプローチにより、開発者は組織全体で部門横断的な連携を行うことができます。その場合、開発者の責任は単にソフトウェアをリリースすることだけにとどまりません。開発者は、ユーザーのために素晴らしい体験を作り出そうとしているチームの一員となります。

こうしたソフトウェア開発アプローチにより、開発者はサイロ化から脱却し、調整を図り、チームワークを促進して、他者からさまざまなことを学ぶ機会を得ることができます。問題解決の形も変わり、単に技術的な問題を解決するのではなく、ユーザーに最も役立つ方法をとるようになります。

このアプローチでは、従業員の関与の度合いを高めるとともに、燃え尽き症候群に関連する閉塞感を防ぐことができる知的な刺激に満ちた環境を作ることができます。

組織にできること

調査結果に基づき、組織には、ユーザーを知ることに時間とリソースを投資することをおすすめします。誰のために作っているのか、そしてユーザーがどのような課題を抱えているのかを把握することに注力しましょう。DORA は、これは価値のある投資になると強く信じています。

ユーザーを決め付けてしまわないようにしましょう。ユーザーをありのままに観察し、質問します。ユーザーの話に基づいて方針転換する謙虚さを持ちましょう。そうすることで、開発者は生産性が向上し、燃え尽き症候群に陥りにくくなり、質の高いプロダクトを提供できるようになります。

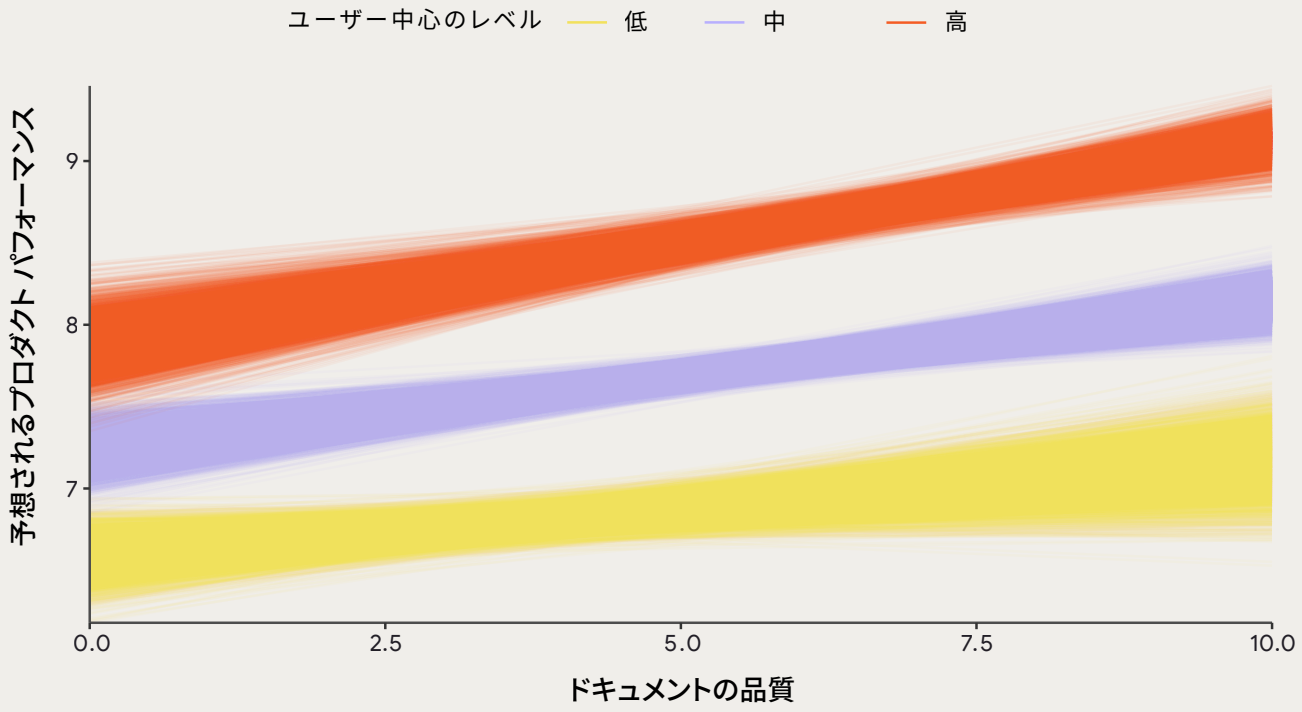
優れたドキュメントとユーザー中心のソフトウェア開発アプローチの組み合わせ

ユーザーを重視するチームは、プロダクトのパフォーマンスが向上します。このユーザー重視の姿勢と質の高い内部ドキュメント環境が組み合わさると、プロダクトのパフォーマンスは一層向上します（図 16 参照）。この調査結果は、技術的能力が組織のパフォーマンスに与える影響はドキュメントによって増大するという現象と似ています[7]。

ドキュメントは、ユーザー シグナルやフィードバックをチーム全体に広め、プロダクト自体に反映させるために役立ちます。

ユーザー シグナルがないと、内部ドキュメントは予想されるプロダクト パフォーマンスに意味のある影響を与えないことがわかっています。しかし、チームに質の高い内部ドキュメントがあれば、そこに含まれるユーザー シグナルがプロダクトのパフォーマンスに大きな影響を与えるでしょう。

2021 年にドキュメントに注目して以来、これまで毎年、質の高いドキュメントのさまざまな影響が確認されています。今年の調査結果では、予想されるプロダクト パフォーマンスに対する内部ドキュメントの影響がリストに加わりました。



このグラフは、最も妥当なパターンの推定を試みたシミュレーションによる、12,000 本の線で構成されています。
図 16: 3 つのレベルのユーザー中心アプローチにおけるプロダクト パフォーマンスとドキュメントの品質

ドキュメントの文化

アジャイル マニフェストは、「包括的なドキュメントよりも機能するソフトウェア」を提唱しています[7]。しかし、質の高いドキュメントが機能するソフトウェアの重要な要素であることが継続的に確認されています。

「包括的なドキュメント」は、ドキュメントを含むさまざまな不健全な手法を言い換えた表現と言えるかもしれません。問題のあるドキュメントとしては、形式のためだけに作成されたドキュメントや、経営陣と従業員の間の不信感を隠すためのドキュメントなどが挙げられます。不健全なドキュメント文化には、作成したドキュメントのメンテナンスや統合を行わないことも含まれます。

そのような場合、質の高いドキュメントに関する尺度のスコアが低くなります。この種のコンテンツは誤った読者に向けて書かれているため、仕事で使おうとしてもあまり役に立ちません。また、ドキュメントが多すぎても、不十分な場合と同様に問題が生じます。

質の高いドキュメントに関する尺度には、ドキュメントの見つけやすさや信頼性といった属性が含まれます。内部ドキュメントの場合、主な読者は同僚か、特定のタスクを達成しようとしている未来の自分自身だということを忘れないようにしましょう[8]。健全なドキュメント文化を持つチームは、こうした読者に資することに焦点を当てています。ここでも、ユーザーに注目することが重要になってきます。

質の高いドキュメントを作成するために DORA で明らかにした以下の方法に沿って取り組むことで、チーム内に健全なドキュメント文化を築くことができます。

- 重要なユースケースを文書化する。
- テクニカル ライティングのトレーニングを受ける。
- ドキュメントを更新する担当者とプロセスを定める。
- チーム内で文書化作業を分担する。
- ソフトウェア開発ライフサイクルの一環としてドキュメントをメンテナンスする。
- 古くなった、または余分なドキュメントを削除する。
- 業績評価や昇進の際、文書化作業を評価する。

優先順位が変わり続けることの危険性

誰もが覚えのある経験です。あなたはここ数か月、新機能の開発に取り組んできました。ユーザーのために必要な機能だと確信しており、集中し、やる気も十分です。しかし突然、あるいは少なくとも突然と感じられるタイミングで、経営陣が組織の優先順位を変更することにしたのです。今や、プロジェクトが中断されるのか、中止されるのか、つぎはぎにされるのか、変更されるのかわかりません。

このようなよくある状況は、従業員や組織にとって重大な意味を持つ可能性があります。ここでは、組織が優先順位を変え続けた場合に何が起きるのかを検証します。

調査結果とその意味

調査結果を総合すると、組織の優先順位が不安定な場合、わずかながら生産性が有意に低下し、燃え尽き症候群が大幅に増加することを示しています。

データによると、こうした燃え尽き症候群の増加を軽減するのは困難です。今回、強いリーダー、優れた内部ドキュメント、ユーザー中心のソフトウェア開発アプローチを持つことで、優先順位の不安定性が燃え尽き症候群に及ぼす影響を打ち消すことができるかどうかを検証しました。

答えは、不可能です。組織がそうした良い特徴をすべて備えていても、優先順位が不安定だと、従業員は燃え尽き症候群に陥るリスクにさらされます。

組織の優先順位の不安定性が従業員のウェルビーイングに悪影響を及ぼす理由

組織の優先順位が不安定だと従業員の燃え尽き症候群が増えるのは、期待される内容が不明確になり、従業員のコントロール感が低下し、仕事量が増加するためであるという仮説を立てました。

なお、優先順位を変えること自体が問題なのではありません。ビジネスの目標やプロダクトの方向性は常に変化するものです。組織の優先順位が柔軟なのは、良いことである場合もあります。

DORA では、優先順位の変更の「頻度」が、従業員のウェルビーイングに悪影響を及ぼすのだと考えています。優先順位の不安定さは、優先順位が頻繁に変更される慢性的なパターンがあることを暗示しています。

数十年にわたる学術研究により、慢性的なストレスは健康とウェルビーイングに悪影響を及ぼすことがわかっています[9]。慢性的なストレスに関する研究と今回の調査結果の間には類似点があります。慢性的な不安定性により、不確実性が高まり、状況をコントロールできるという感覚が弱くなります。この組み合わせから燃え尽き症候群に至ります。

優先順位が安定するとどうなるか

今回の調査結果は少し不可解です。優先順位が安定すると、ソフトウェア デリバリー パフォーマンスが低下することがわかりました。ソフトウェア デリバリーの速度と安定性が低下したのです。

そこで、優先順位が安定している組織ではプロダクトやサービスが概して良好であり、変更があまり頻繁に行われないからではないか、という仮説を立てました。また、優先順位が安定していることで、推奨より少ない回数、大きなバッチでリリースされている可能性もあります。

とはいえ、この調査結果は予想外でした。皆様は、組織の優先順位が安定すると、ソフトウェア デリバリーの速度と安定性が低下するのはなぜだと思われますか？

エンドユーザー向けの AI の構築は優先順位の安定性をもたらすが、ソフトウェア デリバリーの安定性はもたらさない

エンドユーザー向けに AI を活用したエクスペリエンスを取り入れると、組織の優先順位が安定します。これでは AI にお墨付きを与えたように思われるかもしれません。しかし、この調査結果が、AI 自体について何か意味のあることを伝えているとは考えていません。

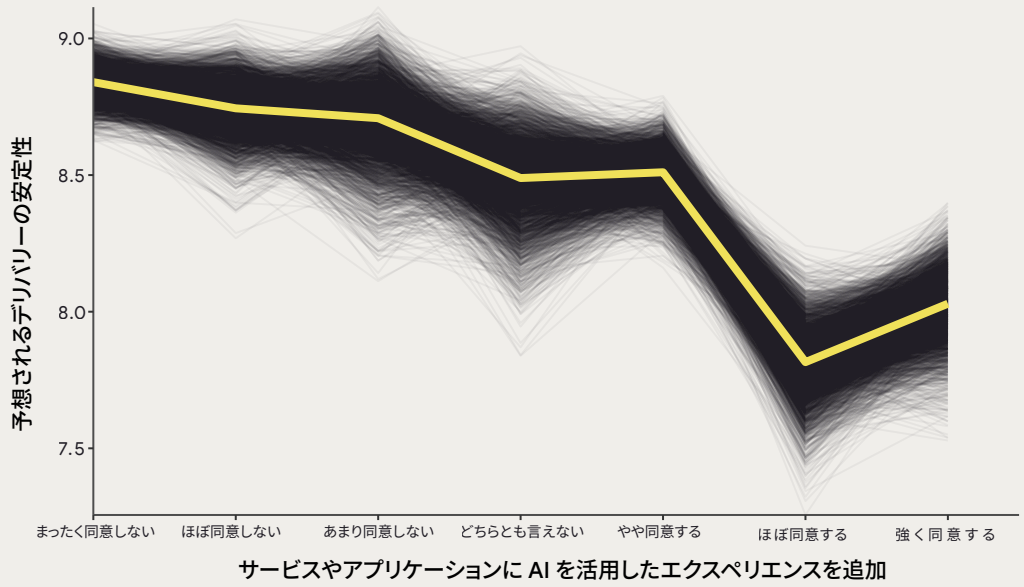
むしろ、取り組みを AI の構築に移行することで、明確性と組織が従うべき指針が得られるのだと考えています。AI ではなくこの明確性が、組織の優先順位の安定化につながります。

ここから、新しいテクノロジーが登場したときに組織に起きることが見てとれるため、これは注目に値します。新しいテクノロジーは変化をもたらします。そして組織には適応するための時間が必要です。

この期間は、リーダーが組織にとって最善の対応を見つけようとするため、優先順位が不安定になりがちです。状況が落ち着き、組織の次のステップが明確になると、優先順位が安定し始めます。

しかし優先順位が安定しても、ソフトウェア デリバリー プロセスがすぐに安定するわけではありません。DORA の分析では、サービスやアプリケーションに AI を活用したエクスペリエンスを追加するように移行する際は、課題と代償が伴うことがわかっています。

移行したチームは、移行していないチームに比べてソフトウェア デリバリーの安定性が 10% も低くなりました。以下に、この課題を図示します。



* 各線は最も妥当な推定を試みたシミュレーション 4,000 件のうちの一つ

図 17: サービスやアプリケーションに AI を活用したエクスペリエンスを追加した場合のソフトウェア デリバリーの安定性

組織にできること

答えは簡単ですが、それほど単純ではなさそうです。調査結果に基づき、組織には、優先順位の安定化に注力することをおすすめします。これは、優先順位の不安定性が従業員の燃え尽き症候群に及ぼす悪影響を打ち消す、確実な方法の一つです。

調査結果によると、優先順位の不安定性が及ぼす悪影響は、優れたリーダー、優れたドキュメント、ユーザー中心のソフトウェア開発アプローチを備えていても相殺されません。このことから、組織が燃え尽き症候群を避けるためにできることは、(1) 優先順位を安定させる、(2) 優先順位の頻繁な変更から従業員が日常的に影響を受けることがないようにする、という方法以外にはあまりないと考えられます。

1. https://www.nngroup.com/articles/bridging-the-designer-user-gap/
2. https://executiveeducation.wharton.upenn.edu/thought-leadership/wharton-at-work/2024/03/creating-meaning-at-work/
3. https://www.apa.org/pubs/reports/work-in-america/2023-workplace-health-well-being
4. https://bigthink.com/the-present/harvard-business-review-americans-meaningful-work/
5. https://hbr.org/2018/11/9-out-of-10-people-are-willing-to-earn-less-money-to-do-more-meaningful-work
6. (P1) など、(P[N]) は、インタビュー参加者の仮名を表しています。
7. https://cloud.google.com/blog/products/devops-sre/deep-dive-into-2022-state-of-devops-report-on-documentation、および「Accelerate State of DevOps Report 2023」(https://dora.dev/research/2023/dora-report)
8. https://agilemanifesto.org/
9. 経営陣、規制機関、監査人など、他の読者も存在します。
10. Cohen S, Janicki-Deverts D, Miller GE.Psychological Stress and Disease.JAMA.2007;298(14):1685–1687.doi:10.1001/jama.298.14.1685

変革を主導する

変革を成功させるには、さまざまな条件が揃っている必要があります。今年の調査では、安定性を優先し、ユーザーを重視して、優れたリーダーを持ち、質の高いドキュメントを作成しているチームが、高いパフォーマンスを発揮していることがわかりました。また、変革を成功させる際の指針となる有益な道筋も明らかになりました。

成功の鍵は、継続的な改善という考え方で変革に取り組むことでした。今回の調査での高パフォーマーは足を引っ張っている変数を理解し、DORA の指標をベースラインとして計画的かつ継続的に改善しています。長期的に成功するにはすべての要所で秀でている必要がありますが、DORA の 10 年にわたる調査から、組織で変革を推進するための具体的かつ効果的な 4 つの方法が明らかになりました。

変革型リーダーシップ

変革的リーダーシップは、リーダーが従業員の価値観や目的意識に訴えかけることで、従業員が高いパフォーマンスを発揮できるように鼓舞し、動機付けて、組織全体の変革を促すモデルです。

そうしたリーダーは以下の要素を通じて、チームが共通の目標に向かって取り組むよう促します[1]。

ビジョン	チームと組織の方向性について明確なビジョンを持っています。
鼓舞するコミュニケーション	チームについて前向きな発言をします。そうすることで、従業員が組織の一員であることを誇りに思い、変化する状況をチャンスに満ちた状況だと捉えることができるよう促します。
知的触発	古い問題を新しい方法で考え、仕事についての基本的な前提を見直すようにチームメンバーに促します。
支えとなるリーダーシップ	行動する前にチームメンバーの個人的感情に配慮します。メンバーの個人的なニーズを尊重した態度で行動します。
個人の評価	チームメンバーが平均より優れた成果をあげた場合は称賛し、業務の質が向上したことを認めます。

今年は、変革的リーダーシップが従業員の生産性向上につながることがわかりました。変革的リーダーシップが 25% 増えると、従業員の生産性が 9% 向上します。

変革的リーダーシップは、生産性の向上以外にも役立ちます。優れたリーダーを持つことは、以下のようなことにもつながります。

- 従業員の燃え尽き症候群の減少
- 仕事の満足度の向上
- チーム パフォーマンスの向上
- プロダクト パフォーマンスの向上
- 組織パフォーマンスの向上

調査から、前述のリーダーシップの質と 2017 年の IT パフォーマンスの間に統計的に有意な関係が認められました。パフォーマンスの高いチームでは、リーダーが 5 つの特性すべてで高いスコアを獲得しており、パフォーマンスの低いチームではスコアが最も低くなりました。さらに、変革的リーダーシップと従業員ネット プロモーター スコア（eNPS、従業員がその会社で働くことを他の人にすすめる可能性）には、強い相関があることがわかりました。

とはいえ、変革的リーダーシップはそれ自体が高パフォーマンスにつながるわけではありません。イネーブラーとして捉えるべきです。

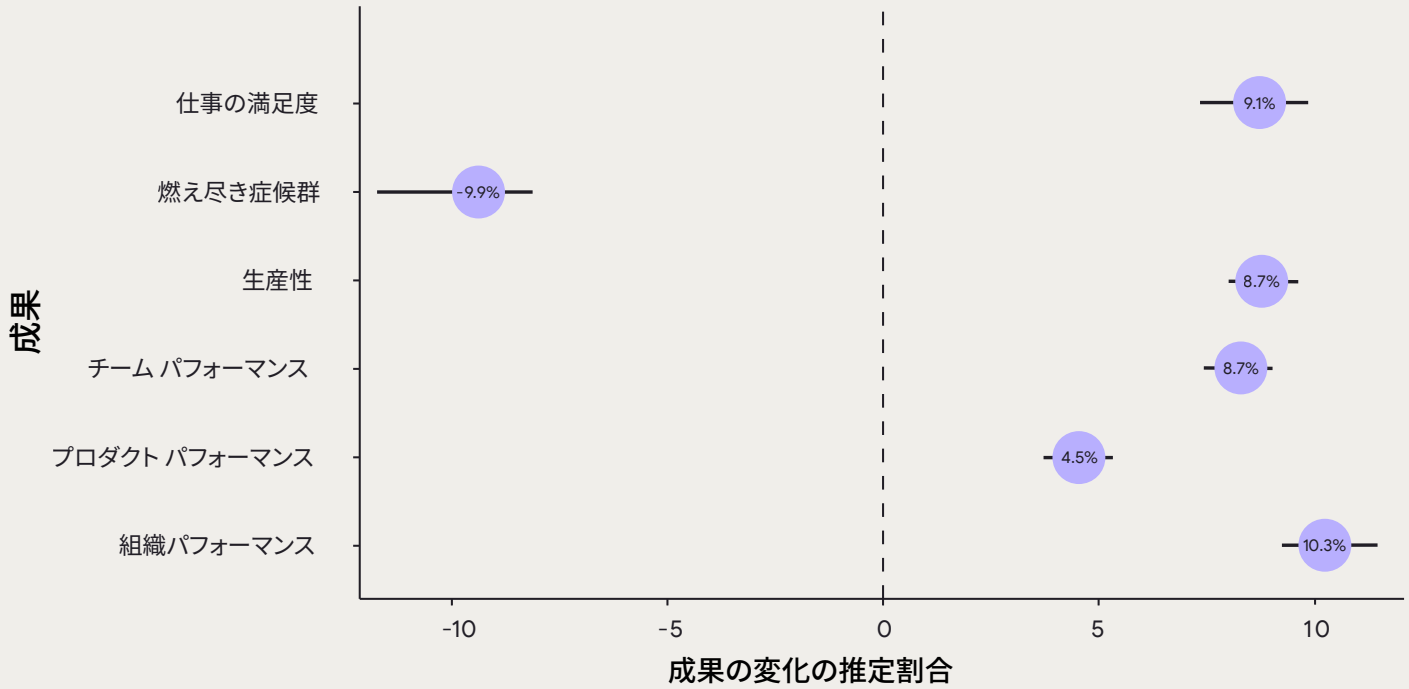
変革的リーダーシップは、技術やプロダクト管理に関する能力と手法を実現するうえで重要な役割を果たします。これは、(1) チームに権限と自主性を持たせること、(2) 問題解決に必要な指標とビジネス インテリジェンスを提供すること、(3) 機能の提供ではなく価値の提供を中心としたインセンティブ構造を築くことによって実現します。

変革には時間がかかり、ツールが必要になります。リーダーシップによって、特に改善のタスクのためにリソースを割り当てる必要があります。優れたリーダーは、改善に必要な時間と資金をチームに提供するうえで重要な役割を果たします。エンジニアは、余暇で新しいことを学んだり自動化したりすることを期待されるべきではありません。そうしたことはスケジュールに組み込む必要があります。

DORA の調査は、IT はコストセンターであるという見方から、IT はビジネスを成功に導く投資であるという見方へ転換するために役立ってきました。DORA は、2020 年には ROI of DevOps ホワイトペーパーを発表しました[2]。その中で、IT の改善に投資することで生み出される潜在的な価値を明らかにできる計算式を紹介しています。

金銭的利益は、この投資から期待できる利益の一つにすぎません。2015 年の調査では、「組織における DevOps への投資は、『組織文化』、『開発、運用、情報セキュリティ チームが Win-Win の成果を達成する能力』、『燃え尽き症候群の度合いの低さ』、『効果的なリーダーシップ』、『継続的デリバリーと無駄のない管理手法の効果的な実践』と強い相関がある」ことを示しました[3]。特に改善のために一定のキャパシティを割くことをおすすめします。

変革的リーダーシップが 25% 増えた場合



点は推定値

エラーバーは 89% 不確実性区間

図 18: 変革的リーダーシップがさまざまな成果に与える影響

ユーザー中心を徹底する

今年の調査では、強いリーダーがいて、ユーザーニーズに対応するソフトウェアの構築に注力している組織は、優れたプロダクトを開発する傾向にあることが示されました。これは強力な組み合わせです。ユーザーをソフトウェア開発の中心に据えると、リーダーは明確なビジョンを持つことができます。

最終的な目標は、作り出したプロダクトをユーザーに気に入ってもらうことです。開発者エクスペリエンスの章で述べたように、ユーザーを重視すると、プロダクトの機能に存在意義が生まれます。開発者は、ユーザー エクスペリエンスの向上に役立つことがわかっていれば、自信を持ってそうした機能を構築できます。

ユーザーニーズを把握してそれに沿うことを強く望み、ユーザー フィードバックの収集、追跡、対応を行うメカニズムを備えたチームは、最も高いレベルの組織パフォーマンスを発揮しています。実際、ユーザーを重視している限り、ソフトウェアの提供速度や安定性のレベルが高くなくても組織は成功できます。2023 年には、ユーザー中心のアプローチを採用しているチームは、そうでないチームと比較して、組織パフォーマンスが 40% 高いことがわかりました[4]。2016 年にも、ユーザー中心のアプローチを採用したチームのほうが組織パフォーマンスが高くなることが明らかになっています。

今年の調査結果も、これまでの調査結果を裏付けるものでした。ユーザーを重視するチームのほうが、優れたプロダクトを生み出します。プロダクトが改善されるだけでなく、従業員の仕事の満足度も高まり、燃え尽き症候群に陥る可能性は低くなります。

迅速で安定的なソフトウェア デリバリーにより、組織は試験的な運用を行って学びを得る機会を頻繁に設けることができます。理想的には、そうしたテストとイテレーションはユーザー フィードバックに基づいて行います。迅速で安定的なソフトウェア デリバリーにより、テストを行い、ユーザーニーズを詳しく把握し、ユーザーニーズが満たされていない場合は迅速に対応することができます。

また、スピードと安定性をソフトウェア デリバリーに組み込むことで、市場の変化や競合他社に対応しやすくなります。

社内の開発者もユーザーであることを忘れないようにしましょう。社内開発者プラットフォーム（IDP）を通じて、組織は開発者に価値を提供し、そこから外部ユーザーや他の内部ユーザーに価値を提供することができます。

調査から、優れた IDP はプロダクトとして開発されており、ユーザー中心を重視して、開発者が独立して作業できるようなエクスペリエンスを提供していることがわかりました。IDP がこのように導入されていると、個人の生産性、チームの生産性、組織パフォーマンスが高くなります。

データに基づいて動く組織になる

成功への進捗状況を可視化する能力はとてつもなく重要です。過去 10 年間にわたり、DORA はデータに基づいて動く組織になることの重要性を訴えてきました。DORA の 4 つの主要指標は[5]、ソフトウェア デリバリー パフォーマンスを測定するための世界標準となりましたが、これは取り組みの一部にすぎません。さらに、組織の改善を推進するために使用できる 30 以上の能力とプロセスを特定しました[6]。

指標の価値は、改善しているかどうかを確認できる点にあります。4 つの主要指標は、組織や事業部門のレベルではなく、アプリケーションやサービスのレベルで使用する必要があります。指標は継続的改善の取り組みを可視化するために使用すべきであり、チーム、ましてや個人を比較するために使用してはなりません。

また、指標は、アプリケーションやサービスのチームの成熟度モデルとして使用してはなりません。低、中、高、エリートのどのパフォーマーであるかは興味をひきますが、これらの呼び名は変革の取り組みの観点ではほとんど価値がないため、注意が必要です。

調査は発展し、進化しているため、4 つの主要指標にとらわれずに考えることをおすすめします。ユーザー フィードバックの指標は、4 つの主要指標と同じくらい重要であることが明らかになっています。これは、ほとんどのチームが速度と安定性を向上させるための実用的な解決策を考案しているからだと考えられます。その結果、パフォーマンスの向上が広まるにつれ、速度と安定性で得られる効果が減っています。

変革を総合的に考えるために、技術指標（4 つの主要指標、信頼性指標など）とビジネス指標を組み合わせたダッシュボードやビジュアリゼーションを作成することをおすすめします。そうすると、トップダウンとボトムアップの変革の取り組みにおけるギャップを埋めるために役立ちます。また、指針、OKR、従業員の目標を IT に対する投資と結びつけるためにも役立ちます。これにより ROI を定量化できます。

指標は、卓越性の要件だと言えます。指標のおかげで意思決定がしやすくなります。定量的、定性的な指標を多く集めるほど、多くの詳細な情報に基づいた意思決定を行えるようになります。人はデータの価値や意味について常に意見を持つものですが、意思決定の根拠としてデータを使用するほうが、意見や直感に頼るよりも望ましいと言えます。

クラウドに全面的に移行するか、データセンターにとどまるか

2018 年以来、DORA は、NIST が定義したクラウド コンピューティングの 5 つの特性（オンデマンド セルフサービス、広範なネットワーク アクセス、リソースプール、迅速な拡張、測定可能なサービス（別名「柔軟なインフラストラクチャ」））と[7]、組織パフォーマンスの関係について調査を行ってきました。成功しているチームは、成功していないチームよりも柔軟なインフラストラクチャを活用する傾向にあります。

昨年の調査では、このトピックに関してこれまでで最も印象的な情報が得られました。5 つの特性を活用せずにクラウドを使用することは有害な可能性があり、組織パフォーマンスの低下につながるのです。組織は、アプリケーションやサービスを抜本的に変革するつもりがないのであれば、データセンターにとどまるほうがよいかもしれません。もちろん、変革のためには単にツールやテクノロジーを導入するだけでなく、多くの場合、アプリケーションの設計、構築、デプロイ、運用に関してまったく新しいパラダイムを導入する必要があります。大規模な変更を行う際は、少数のサービスで始めるほうが簡単です。チームや組織が前進しながら学習と改善を行えるように、反復的なアプローチを採用することをおすすめします。

まとめ

過去 10 年間にわたって一貫して確認されてきたのは、変革が成功の要件だということです。変革は最終目標ではなく、継続的な改善の取り組みですが、このことを多くの組織が誤解しています[8]。調査から、継続的な改善を行っていない企業は後れを取っていることが明らかになっています。逆に言うと、継続的な改善の考え方を採用している企業は、最高水準の成功を収めています。

この取り組みを進める中で、多少の痛みや困難に見舞われる可能性があります。調査によると、DevOps[9]、SRE[10]、そして今年はプラットフォーム エンジニアリングで、最初にパフォーマンスが低下した後、大幅に向上していました（「J 曲線」ともいいます）。これは正常です。継続的な改善を行っていれば状況は良くなり、最終的には初めよりもはるかに良い状態になります。

終わりのない取り組みという考え方に、気が遠くなるかもしれません。完璧な変革を計画、設計しようとして行き詰まることはよくあります。成功の鍵は、腕をまくって、ただ作業に取り掛かることです。組織やチームの目標は、単に、昨日より少しでも良くすることです。DORA の過去 10 年間の調査の目標は、皆様が継続的に改善して向上していけるよう支援することであり、それは今後も変わりません。

1. Dimensions of transformational leadership: Conceptual and empirical extensions - Rafferty, A. E., & Griffin, M. A.
2. DevOps 変革の ROI - https://dora.dev/research/2020/
3. 2015 年の State of DevOps Report https://dora.dev/research/2015/2015-state-of-devops-report.pdf#page=25
4. 2023 年の Accelerate State of DevOps Report - https://dora.dev/research/2023/dora-report/2023-dora-accelerate-state-of-devops-report.pdf#page=17
5. DORA の 4 つの主要指標 https://dora.dev/guides/dora-metrics-four-keys/
6. DORA の能力とプロセス https://dora.dev/capabilities/
7. NIST が定義したクラウド コンピューティングの 5 つの特性 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
8. 継続的な改善の取り組み https://cloud.google.com/transform/moving-shields-into-position-organizing-security-for-digital-transformation
9. 2018 年の Accelerate State of DevOps Report https://dora.dev/research/2018/dora-report/
10. 2022 年の State of DevOps Report https://dora.dev/research/2022/dora-report/

DORA の 10 年

経緯

DevOps ムーブメントは、2009 年、トピックが関連しているものの、それ以外の部分では連携していない 2 つのイベントから生まれました。まず、John Allspaw 氏と Paul Hammond 氏が、同年 6 月に開催された Velocity カンファレンスで「10 deploys per day: Dev & ops cooperation at Flickr」と題した講演を行います[1]。その数か月後、Pactrick Debois 氏がボランティアの主催者チームを率いて、ベルギーのゲントで最初の DevOpsDays イベントを開催しました[2]。

その発展に DevOps コミュニティの注目が集まるまで時間はかかりませんでした。Puppet Labs の Alana Brown 氏は 2011 年、DevOps について詳しく知るための調査を実施します。この調査により、「DevOps で仕事を進めることが、IT における新しいビジネスの方法として台頭している」ことが確認されました。

このムーブメントが新たな業界や組織に拡大していく中、Alana Brown 氏はこの成功に基づき、IT Revolution Press と連携して別の調査を 2012 年に実施。その調査結果を 2013 年の State of DevOps Report で発表しました[3]。

翌年には調査チームに Nicole Forsgren 博士が加わり、プログラムに科学的な厳密さをもたらしました。2014 年の State of DevOps Report では[4]、ソフトウェア デリバリー パフォーマンスと組織パフォーマンスの関係について、「IT チームのパフォーマンスが高い上場企業は、IT 組織のパフォーマンスが低い企業よりも、3 年間の時価総額成長率が 50% 高い」ことが明らかになりました。

年次レポートのトレンドは 2016 年までに確立し、Forsgren 氏、Jez Humble 氏、Gene Kim 氏は DevOps Research and Assessment（DORA）を設立しました。この年の State of DevOps Report では、DevOps の手法を導入したチームによる投資を測定するための計算式が紹介されています。この成果は、2020 年に発表された DevOps 変革の ROI に関するホワイトペーパーにつながりました[5]。

『Accelerate: The science behind devops: Building and scaling high performing technology organizations』は[6]、Forsgren 氏、Humble 氏、Kim 氏が執筆し 2017 年に IT Revolution Press から発行されました。この書籍は、改善を推進する能力に注目し、調査プログラムの初期の内容をまとめています。

企業としての DORA は、2018 年に独自のレポート「Accelerate State of DevOps: Strategies for a New Economy」を公開しました[7]。Puppet のチームはこの年から DORA とは別に独自のレポート シリーズを公開し続けています[8]。

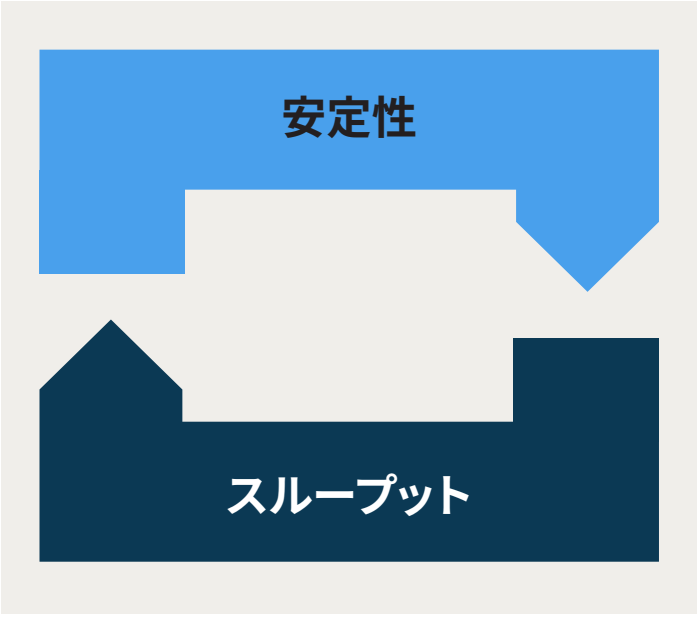
2018 年、DORA は Google Cloud に買収され[9]、プラットフォームに依存しない科学的な調査を続けています。DORA Report は今年で 10 年目を迎えます[10]。調査結果を共有できることを嬉しく思います。お読みいただきありがとうございます。

DORA の主な分析情報

チームは安定性のためにスピードを犠牲にする必要はない

テクノロジー主導型チームには、現状を評価し、改善の優先順位を決め、進捗を検証するために、パフォーマンスを測定する方法が必要です。DORA は、ソフトウェア デリバリー プロセスの成果を効果的に測定できる 4 つのソフトウェア デリバリー指標（4 つの主要指標）を明らかにし、検証してきました。ソフトウェア デリバリー パフォーマンスに関するこれらの尺度は、業界標準となっています。

調査から、変更のスループットと安定性は連動する傾向にあることがわかりました。あらゆる業種において、両方を高い水準で達成しているチームが確認されています。



チームが 4 つの主要指標を測定する方法はたくさんある

- チーム ミーティング中の会話や振り返りを通じて
- DORA クイック チェック（https://dora.dev/quickcheck）
- ソフトウェア エンジニアリング インテリジェンス（SEI）カテゴリの商用ツールやソース アベイラブル ツール[11]
- チームが使用している特定のツール向けに構築された特注のインテグレーション

ソフトウェア デリバリー パフォーマンスと運用パフォーマンスが組織パフォーマンスを促進する

DORA はソフトウェア デリバリー パフォーマンスを測定するために 4 つの主要指標を使用しています。DORA は 2018 年に初めて運用パフォーマンスを調査しました。これは、ソフトウェア プロダクトまたはサービスに関する約束や主張を行ったり、それを守ったりする能力を測定するものです。

ソフトウェア デリバリー パフォーマンスと運用パフォーマンスの両方が一体となって、組織パフォーマンスと従業員のウェルビーイングを促進すると、最高の結果が得られます。

テクノロジー主導型チームの実務担当者は、アプリケーション ユーザーの信頼性に関する期待に応えつつデリバリー プロセスにおける摩擦を減らすことの重要性を認識しています。

文化は成功のための最重要事項

組織の文化は、最も明らかなパフォーマンスの予測因子の一つです。学びと連携を促進する強い信頼に基づく文化の威力が、継続的に確認されています。たとえば 2022 年の調査では、組織のアプリケーション開発におけるセキュリティ対策について、文化が最大の予測因子であることが示されました[12]。

文化は調査のあらゆる面で影響を与えており、多面的で常に変化しています。そのため、Westrum の「Typology of Organizational Culture」などの研究から着想を得て[13]、長年にわたりさまざまな尺度を使用してきました。ウェルビーイングの尺度としては、燃え尽き症候群、生産性、仕事の満足度などがあります。

継続的に改善して向上

DORA ではチームに対し、継続的に改善して向上してくための目標を定めるよう促しています。改善を促進するには、継続的な改善の考え方と手法が必要です。そのためには、現在の状況を評価し、改善作業に優先順位を付ける方法や、進捗を測定できるフィードバック メカニズムが必要です。

改善への実験的アプローチには成功と失敗の両方が伴いますが、どちらの場合もチームは得られた教訓をもとに有意義な行動をとることができます。

今後の 10 年

総じて、この 10 年間、私たちは互いに多くのことを学んできました。年次アンケートへのご協力、DORA Community of Practice[14]へのご参加、そして DORA をそれぞれの状況に合わせてご活用いただいていることに、感謝申し上げます。

テクノロジーに関する状況が発展を続ける中、DORA は、テクノロジー主導型のチームや組織が成功するための能力と手法を継続的に調査していきます。また、テクノロジーの人間的側面を優先するとともに、皆様の取り組みの指針となるような、プラットフォームに依存しない調査を発表していきます。

過去の分析情報の多くは、新しいテクノロジーや手法へのアプローチに今でも十分に役立つと考えていますが、今後も皆様のご協力を得て新しい分析情報を発見していくことを楽しみにしています。

DORA は、DevOps ムーブメントの一環として常に掲げられてきた基本原則、つまり文化、連携、自動化、学習、そしてビジネス目標を達成するためのテクノロジーの活用に、忠実に取り組んでいます。DORA のコミュニティと調査は、「DevOps」とはあまり縁のない方々を含め、さまざまな立場の方のご意見から恩恵を受けています。いつかは、「DevOps」という言葉が脚光を浴びることもなくなるでしょう。

今年のレポートは、特に AI の活用と影響を重視しています。前述のとおり、導入が増えており、この分野は試行錯誤の余地が多くあります。DORA は今後もこの分野や他の新しいテクノロジーと手法について、調査を行っていきます。過去の調査結果と新しい調査結果を併用することで、導入を促進するとともに、チーム全員のデベロッパー エクスペリエンスの改善にお役立てください。

1. スライド - https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr、動画 - https://www.youtube.com/watch?v=LdOe18KhtT4
2. https://legacy.devopsdays.org/events/2009-ghent/
3. https://www.puppet.com/resources/history-of-devops-reports#2013
4. 2014 年の State of DevOps Report - https://dora.dev/research/2014/
5. DevOps 変革の ROI - https://dora.dev/research/2020/
6. Forsgren, Nicole, Jez Humble, and Gene Kim.2018.Accelerate: The Science Behind DevOps : Building and Scaling High Performing Technology Organizations.IT Revolution Press.
7. Accelerate State of DevOps: ニュー エコノミーに向けた戦略 - https://dora.dev/research/2018/dora-report/
8. https://www.puppet.com/resources/history-of-devops-reports#2018
9. https://dora.dev/news/dora-joins-google-cloud
10. DORA が設立されたのは数年後ですが、Forsgren 博士がプログラムに参加した 2014 年を最初の DORA レポートの年だと考えています。2020 年はレポートを発表していないため、2024 年がレポートの 10 年目となります。
11. https://dora.dev/resources/#source-available-tools
12. 2022 年の Accelerate State of DevOps Report - https://dora.dev/research/2022/dora-report/
13. Ron Westrum、A typology of organisation culture、BMJ Quality & Safety 13、no. 2 (2004 年)、doi:10.1136/qshc.2003.009522
14. https://dora.community

最後に

DORA は過去 10 年にわたり、調査、見識、情報の信頼できるソースとしての地位を確立してきました。業界でプラットフォーム エンジニアリングや AI といった新しい手法やテクノロジーの導入が続く中、DORA は皆様とともに、チームの改善に役立つ取り組みを調査していきます。今後とも DORA をよろしくお願いいたします。

調査の再現

今年のレポートにおける調査分野や調査結果は複雑であり、不明確な場合や、矛盾している場合があります。そのため、DORA の調査を再現することをおすすめします。1 つのチームや組織に注目することで、理解を深める多くの機会が開かれます。

組織内で実験を行う

DORA の調査結果は、次回の実験のための仮説として利用できます。チームの業務の進め方について詳しく調べ、DORA の研究プログラムの調査結果を参考にして、改善が必要な分野を明らかにしましょう。

組織内でアンケートを実施する

このレポートと今年の調査で使用した質問を参考にして[1]、独自の社内アンケートを作成しましょう。アンケートには、対象者に合わせて微調整した質問を組み込むことができます[2]。DORA の調査の実施方法について詳しくは、手法の章をご覧ください。調査結果を実践に移すことに焦点を当てるようにしてください。

学んだことを共有する

実験を通じて学んだことを組織全体に広めましょう。共有する方法は、大勢に向けた公式のレポートや、非公式の実践コミュニティ、同僚との雑談など、さまざまです。さまざまなアプローチを試し、ご自身の状況や文化に最適な方法を見つけてください。これも実験のプロセスと言えます。

調査の活用方法

DORA コミュニティ（https://dora.community）に参加して、経験を共有し、他者から学び、継続的改善に取り組んでいる仲間からヒントを得ましょう。

DORA COMMUNITY

1. 2024 年のアンケート https://dora.dev/research/2024/questions/
2. ソフトウェア企業内で実施した DORA アンケートの経験 - https://www.infoq.com/news/2024/08/dora-surveys-software-company/

謝辞

DORA レポートは、今年で 10 年目という特別な節目を迎えました。この取り組みに参加し DORA とともに歩みを進めてきた研究者、専門家、実務担当者、リーダー、変革エージェントの皆様のご尽力に感謝いたします。

Puppet Labs と IT Revolution Press が最初の State of DevOps Report を発行してから、長い道のりを歩んできました。この道を切り拓いた DORA の創設者に心から感謝します。この間に世界がどれほど変わったか、我々がどれほど学んだかを振り返ると、目を見張るものがあります。

今年の発表にご協力いただいた皆様に改めて感謝いたします。業界の慣行を導き、影響を与えるのは大変なことであり、皆様の貢献は非常に貴重です。

初期の頃から今の刺激的な AI の時代まで、この取り組みに参加していただいたすべての方に感謝します。この取り組みは、皆様の見識や支援の賜物です。次の 10 年も発見と協力に満ちたものになりますように。

DORA レポートチーム

James Brookbank

Kim Castillo

Derek DeBellis

Benjamin Good

Nathen Harvey

Michelle Irvine

Amanda Lewis

Eric Maxwell

Steve McGhee

Allison Park

Dave Stanke

Kevin Storer

Daniella Villalba

編集者

Seth Rosenblatt

ローカライズ ボランティア

Andrew Anolasco

Mauricio Meléndez

Marie-Blanche Panthou

Miguel Reyes

Yoshi Yamaguchi

Jinhong Yu

DORA ガイド

Lisa Crispin

Steve Fenton

Denali Lumma

Betsalel (Saul) Williamson

分野のアドバイザー、専門家

John Allspaw

Birgitta Böckeler

Sander Bogdan

Michele Chubirka

Thomas De Meo

Jessica DeVita

Rob Edwards

Nicole Forsgren 博士

Gene Kim、IT Revolution

Laura Maguire 博士

James Pashutinski

Ryan J. Salva

Majed Samad

Harini Sampath

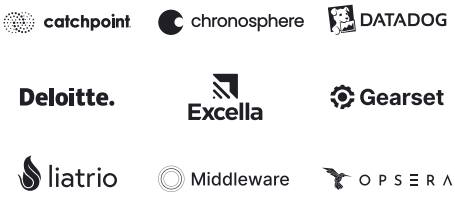
Robin Savinar

Sean Sedlock

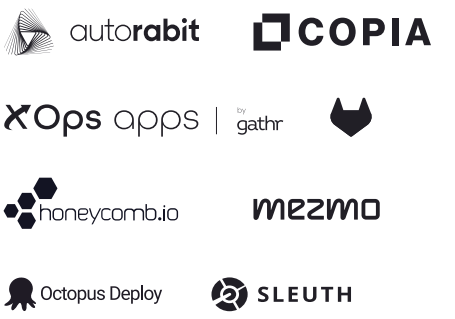
Dustin Smith

Finn Toner

Gold スポンサー



Silver スポンサー



著者

Derek DeBellis

Google の定量的ユーザー エクスペリエンス リサーチャーである Derek は、DORA の主任調査員です。Derek は、アンケート調査やログ分析のほか、プロダクトや機能がユーザーに独自の大きな価値を提供していることを示すコンセプトの測定方法の考案に注力しています。また、人と AI の相互作用、COVID-19（新型コロナウイルス感染症）が禁煙に及ぼす影響、NLP エラーを想定した設計、プライバシーの考察における UX の役割、チーム文化、従業員のウェルビーイングや生産性と AI の関係に関する記事を公開しています。現在は、信条と力の伝播のシミュレーション方法を探る課外研究に取り組んでいます。

Kevin M. Storer

Kevin M. Storer 博士は Google の開発者エクスペリエンス リサーチャーであり、DORA チームでは質的調査のリーダーを務めています。ソフトウェア エンジニアリングの実務経験と、大学院での社会科学と人文科学にまたがる学際的トレーニングを活かし、2015 年以来、幅広い問題背景、参加者プロフィール、調査手法にわたって、ソフトウェア開発者に関する人間中心の研究を主導しています。Kevin の研究は、AI、情報検索、組み込みシステム、プログラミング言語、ユビキタス コンピューティング、インタラクション デザインのトピックで、一流の科学誌に掲載されています。

Amanda Lewis

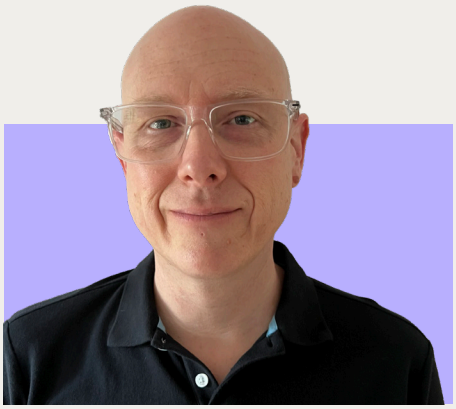
DORA.community の開発リードである Amanda Lewis は、Google Cloud のデベロッパーリレーションズ エンジニアです。Amanda はそのキャリアを通じて、開発者、運用者、プロダクト マネージャー、プロジェクト マネージャー、リーダーの関係構築に携わってきました。また、e コマース プラットフォーム、コンテンツ マネジメント システム、オブザーバビリティ ツールの開発チームで働き、開発者をサポートしてきました。このような関係と対話は、顧客満足度や業績の向上につながります。Amanda は経験と共感力を活かし、チームがソフトウェア デリバリーと AI の手法を理解し実践するための支援を行っています。

Benjamin Good

Ben Good は Google のクラウド ソリューション アーキテクトです。クラウド テクノロジーと自動化を通じてソフトウェア デリバリーの手法を改善することに情熱を注いでいます。またソリューション アーキテクトとして、アーキテクチャのガイダンスの提供、技術ガイドの公開、オープンソースへの貢献を通じて、Google Cloud のお客様の問題解決を支援しています。Google に入社する前は、米国コロラド州デンバーおよびボルダー地域の複数の企業で、DevOps の手法を導入しつつクラウド運用を担当していました。

Daniella Villalba

Daniella Villalba は Google のユーザー エクスペリエンス リサーチャーです。Daniella は開発者が幸福になり生産的になる要素を知るためにアンケート調査を行っています。Google に入社する前は、瞑想トレーニングの利点や、大学生の体験に影響を及ぼす心理社会学的な要素について研究していました。Daniella はフロリダ国際大学から、実験心理学で博士号を授与されています。



Eric Maxwell

Eric Maxwell は Google の DevOps トランスフォーメーションの取り組みでリーダーを務めており、価値の迅速な提供による事業の改善について、世界のトップ企業に助言を行っています。Eric は、キャリアの前半を現場エンジニアとして過ごす中でさまざまな自動化に取り組み、他の実務者への共感を培ってきました。Eric は、Google の Cloud Application Modernization Program（CAMP）の共同創設者で、DORA チームのメンバーです。Google に入社する前は、Chef Software のエキサイティングな環境で同僚とともに優れたプロダクトを生み出していました。

Kim Castillo

Kim Castillo は Google のユーザー エクスペリエンス プログラム マネージャーです。Kim は DORA を支える部門横断的な取り組みを率い、2022 年以来、研究業務や本レポートの公開を監督しています。Google Cloud の Gemini の UX リサーチにも携わっています。Google に入社する前は、テクノロジー分野でキャリアを積み、技術プログラム管理やアジャイル コーチングに携わりました。Kim のキャリアの根底にあるのは、母国フィリピンにおける超法規的殺人、都市貧困層対象の開発事業、コミュニティのレジリエンスをテーマとした、心理社会学的研究です。

Michelle Irvine は Google のテクニカル ライターで、ドキュメント作成などのテクニカル コミュニケーションに関する研究を行っています。Google に入社する前は教育系の出版業務に携わり、また、物理シミュレーション ソフトウェアのテクニカル ライター職に従事していました。Michelle はウォータールー大学から物理学の理学士号と、レトリックおよびコミュニケーション設計の修士号を授与されています。

Michelle Irvine

Google Cloud で DORA チームを率いる Nathen Harvey は、素晴らしい組織、チーム、オープンソース コミュニティから学び、それを共有してきました。複数の DORA レポートの共著者であり、『97 Things Every Cloud Engineer Should Know』(O'Reilly 発行、2020 年) には寄稿者および編集者として貢献しました。

Nathen Harvey

ユーザー属性と企業特性

アンケートの対象者

DORA の研究プログラムは、パフォーマンスの高いテクノロジー主導型組織の能力、手法、尺度について 10 年にわたって調査を行ってきました。その中で、さまざまな業界のあらゆる規模の組織で働く 39,000 人超のプロフェッショナルの方々にお話を伺いました。皆様のご協力に感謝いたします。今年は、全世界から各種の業界に属する約 3,000 人の現役のプロフェッショナルに経験を共有していただき、パフォーマンスの高いテクノロジー主導型組織の原動力となる要素について理解を深めることができました。

今年のユーザー属性と企業特性に関する質問では、Stack Overflow による研究を活用しました。

2023 Stack Overflow Developer Survey には 9 万人を超える回答者が参加しました[1]。このアンケートはさまざまな理由からすべての技術者を対象とはしませんでしたが、開発者界の国勢調査に限りなく近いものになりました。

アンケートから得られる母集団を意識することで、データ内の回答バイアスを見つけ、調査結果をどこまで一般化すべきかを検討できます。さらに、Stack Overflow Developer Survey で尋ねたユーザー属性と企業特性に関する質問は、内容が十分に練られており、借用する価値があります。

手短に言えば、DORA のサンプルと Stack Overflow のサンプルの間に大きな相違はありません。つまり、DORA のサンプルは母集団を反映していると考えて間違いないでしょう。

業種

アンケートの回答者に、所属組織が主に事業を展開している業種を 12 種類の中から選んでもらいました。業種で特に多かった回答は、テクノロジー（35.69%）、金融サービス（15.66%）、小売 / コンシューマー / e コマース（9.49%）でした。

業種	回答者の割合
テクノロジー	35.69%
金融サービス	15.66%
小売 / コンシューマー / e コマース	9.49%
その他	5.94%
工業と製造業	5.49%
ヘルスケア、医薬品	4.60%
メディア、エンターテイメント	4.26%
政府、行政機関	3.89%
教育	3.66%
エネルギー	3.03%
保険	2.39%
非営利団体	1%

従業員数

アンケートの回答者に、所属組織の従業員数を 9 つの枠から選んでもらいました。従業員数で特に多かった回答は、10,000 人以上（24.10%）、100〜499 人（18.50%）、1,000〜9,999 人（15.60%）でした。

組織の規模	割合
1	2.0%
2〜9	3.2%
10〜19	4.3%
20〜99	14.5%
100〜499	18.5%
500〜999	11.2%
1,000〜4,999	15.6%
5,000〜9,999	6.7%
10,000 以上	24.1%

障がい

障がいについては、Washington Group Short Set の基準に従い[2]、6 つの観点で識別しました。障がいについての質問を行うのは今年で 5 年目です。障がいがあると回答した人の割合は、2023 年は 6%、2024 年は 4% と、2022 年の 11% から減少しています。

障がい	回答者の割合
障がいのいずれもあてはまらない	92%
少なくとも 1 つの障がいに当てはまる	4%
回答しない	4%

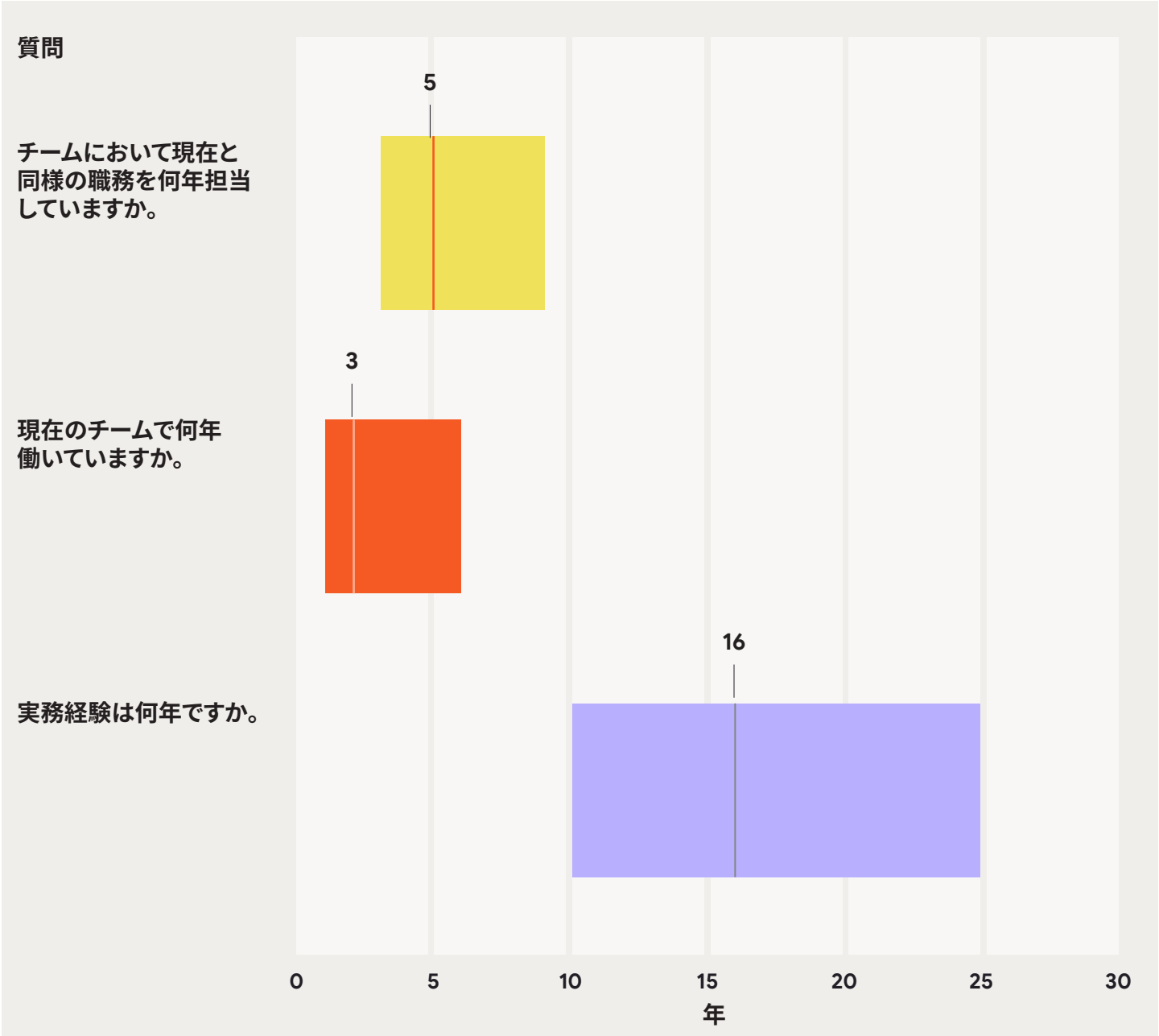
性別

アンケートの回答者に対し、性別を尋ねました。83% が男性、12% が女性、1% が自己決定と回答し、4% は回答を避けました。

性別	割合
男性	83%
女性	12%
自己決定	1%
回答しない	4%

経験

アンケートの回答者に対し、職務とチームでの経験年数を尋ねました。中央値で見ると、実務経験年数は 16 年、現在の職務での経験年数は 5 年、現在のチームでの経験年数は 3 年でした。

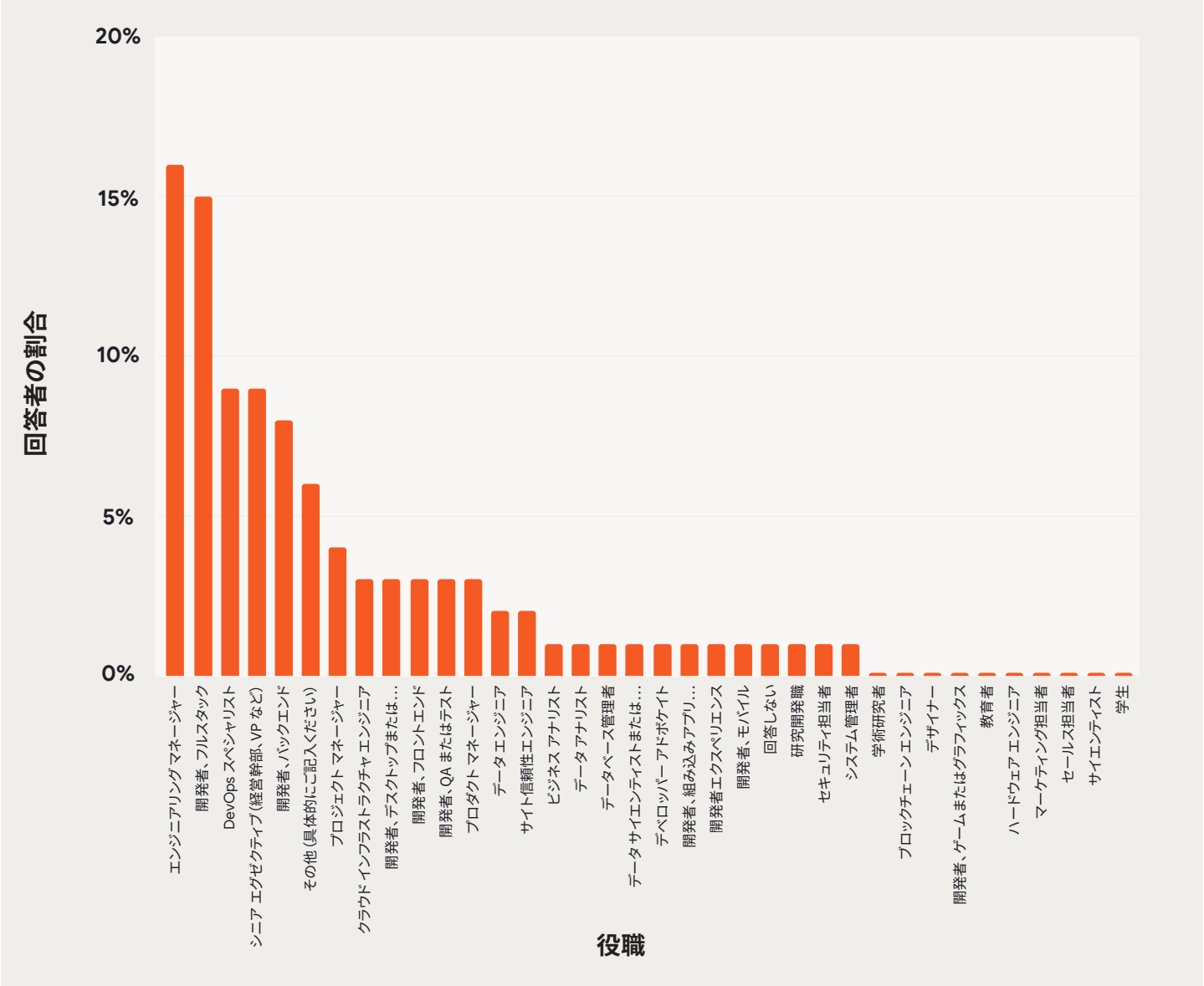


ボックスの幅は 25〜75 パーセンタイルを表す。ボックスを分割している線は中央値を表す。

職務

分析では、回答者の割合が少ない職務も取りこぼさないようにするために、一部の職務をグループにまとめました。その他のカテゴリでは、次のように十分な数の回答者がありました。

- 開発者（回答者の 29%）
- マネージャー（回答者の 23%）
- シニア エグゼクティブ（回答者の 9%、2023 年から 33% 増）
- 分析職（回答者の約 5%）



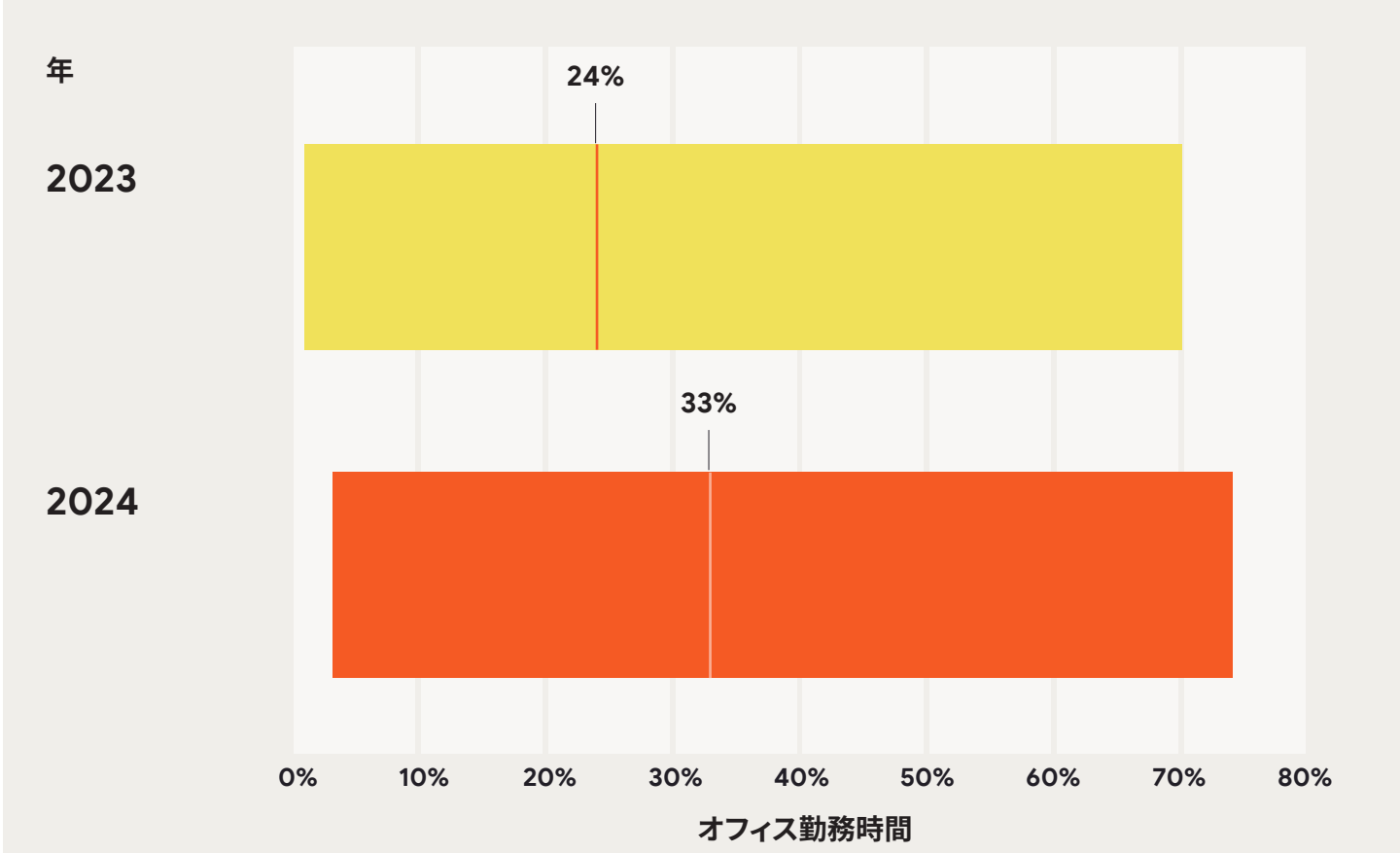
雇用形態

アンケートの回答者に対し、現在の雇用形態を尋ねました。回答者の大多数（90%）は、組織のフルタイム正社員でした。

雇用形態	割合
フルタイム契約社員	6%
フルタイム正社員	90%
パートタイム契約社員	1%
パートタイム正社員	2%

勤務場所

オフィス回帰（RTO）が今年も進んだにもかかわらず、特に分布のテールにかけて、昨年のパターンがほぼ維持されています。中央値が 37.5% 増加したことは、ハイブリッドな勤務形態や、少なくとも定期的な出社が一般的になってきたことを示唆しています。



ボックスの幅は 25〜75 パーセンタイルを表す。ボックスを分割している線は中央値を表す。

国

回答者は 104 か国にわたっています。喜ばしいことに、いつも世界中の方々からアンケートにご協力いただいています。皆様、ありがとうございます。

国

米国	イタリア	シンガポール	アイスランド	ルクセンブルク	グアテマラ
英国	スイス	アルバニア	イラン	ニカラグア	香港（特別行政区）
カナダ	アルゼンチン	ジョージア	ヨルダン	パキスタン	マルタ
ドイツ	メキシコ	ギリシャ	ケニア	ペルー	モーリシャス
日本	ポルトガル	フィリピン	サウジアラビア	韓国	モロッコ
インド	オーストリア	ハンガリー	スロバキア	スリランカ	ネパール
フランス	ルーマニア	セルビア	スロベニア	チュニジア	パラグアイ
ブラジル	フィンランド	アフガニスタン	タイ	アンドラ	スワジランド
スペイン	トルコ	アルジェリア	ウズベキスタン	バルバドス	シリアアラブ共和国
オーストラリア	ブルガリア	エジプト	アンゴラ	ベリーズ	台湾
オランダ	アイルランド	インドネシア	アルメニア	ベナン	マケドニア旧ユーゴスラビア共和国
中国	イスラエル	ロシア連邦	ボスニア ヘルツェゴビナ	ボリビア	トリニダードトバゴ
スウェーデン	ベルギー	ウクライナ	ドミニカ共和国	ブルキナファソ	ウルグアイ
ノルウェー	チリ	ベトナム	エクアドル	コモロ	ベネズエラ ボリバル共和国
ニュージーランド	コロンビア	バングラデシュ	エストニア	コートジボワール	
ポーランド	チェコ共和国	ベラルーシ	カザフスタン	エルサルバドル	
南アフリカ	マレーシア	コスタリカ	ラトビア	エチオピア	
デンマーク	ナイジェリア	クロアチア	リトアニア	ガンビア	

人種や民族

アンケートの回答者に対し、人種や民族を尋ねました。回答者で特に多かったグループは、白人（32.4%）、ヨーロッパ系（22.7%）でした。

人種や民族	割合
白人	32.4
ヨーロッパ系	22.7
アジア系	9.9
北米系	4.6
インド系	4.1
回答しない	4.1
ヒスパニック、ラテン系	3.5
南米系	3.2
東アジア系	2.5
アフリカ系	1.8
南アジア系	1.7
多民族	1.5
自由回答	1.5
東南アジア系	1.4
黒人	1.3

人種や民族	割合
中東系	1.3
2 つの人種の混合	0.4
中米系	0.4
わからない	0.4
北アフリカ系	0.4
カリブ諸国系	0.2
中央アジア系	0.2
南アジア系	1.7
民族宗教グループ	0.2
太平洋諸島系	0.2
先住民族（ネイティブ アメリカン、オーストラリア先住民など）	0.1

1. https://survey.stackoverflow.co/2023/
2. https://www.washingtongroup-disability.com/question-sets/wg-short-set-on-functioning-wg-ss/

手法

手法はレシピのようなものです。手法は、私たちが行った作業を再現し、DORA のデータ生成および分析方法によって価値ある情報がもたらされるのかどうかを判断していただくために役立ちます。スペースの関係ですべてはご紹介できませんが、検討作業の参考にしていただければ幸いです。

アンケートの作成

質問の選択

ある質問をアンケートに追加するかどうかを検討する際は、以下の点を考慮します。

その質問は…

- **結果を過去の取り組みと結びつけられるように設定されているか**
- **業界が達成しようとしている成果を捉えているか（高いチーム パフォーマンスなど）**
- **業界がリソースの投入を検討している機能を捉えているか（AI など）**
- **各自が目標を達成するために役立つと思われる能力を捉えているか（質の高いドキュメントの作成など）**
- **サンプルの代表性を評価するために役立つ内容か（職務、性別など）**
- **バイアス発生を避けるために役立つ内容か（コーディング言語、職務など）**
- **回答者の大多数が一定程度以上の精度で回答できる内容か**

DORA は、文献を調べ、DORA コミュニティと関わり、認知面接を行い、並行して質的調査を実施し、対象分野の専門家の協力を仰ぎ、チーム ワークショップを開催して、質問をアンケートに追加するかどうかを判断する際の参考にしています。

アンケート体験

DORA は、アンケートにおけるユーザビリティの向上に細心の注意を払っています。認知面接やユーザビリティ テストを行い、アンケートが一定の仕様を満たしていることを確認しています。

- **アンケートの所要時間が平均して短いこと**
- **アンケートの理解度が高いこと**
- **労力ができるだけかからないこと（コンセプトの技術的な性質を考えると大きな課題です）**

データ収集

ローカライズ

毎年、世界中の人がアンケートに回答してくださっています。今年はアンケートを英語、スペイン語、フランス語、ポルトガル語、日本語、簡体中国語にローカライズすることで、より多くの人にアンケートに参加していただけるようにしました。

アンケートの回答の収集

回答者の募集には複数のチャネルを利用しました。チャネルはオーガニックとパネルの 2 つに分類されます。

オーガニック アプローチでは、ソーシャルのあらゆる手段を利用して、アンケートを実施することを伝えます。ブログ投稿や、メール キャンペーンを行います。また、ソーシャル メディアに投稿して、コミュニティのメンバーにも拡散を依頼します（スノーボール サンプリング）。

パネル アプローチは、オーガニック チャネルを補うために行います。過小評価 / 低代表（underrepresented）グループの人を広範な技術コミュニティで募集し、特定の業界や組織から十分な回答を得られるよう努めます。

つまり、オーガニック アプローチでは行えない、募集対象の一定のコントロールを行います。パネル アプローチには、単に十分な数の回答者を確保する役割もあります。オーガニック アプローチでは、実施予定の分析に必要な数の回答が得られるかどうかがわからないためです。今年は、分析を実施するために十分な数の回答がオーガニック アプローチで得られ、パネル アプローチは参加者グループのバランスをとるために役立ちました。

アンケート フロー

今年は質問したいことがたくさんありましたが、時間が十分ではありませんでした。取り得る方法は次のとおりです。

- **非常に長いアンケートを作成する**
- **注目する分野を絞る**
- **異なるトピックにランダムに参加者を割り振る**

関心対象のどの分野もあきらめられなかったため、参加者を 3 種類のフローのいずれかにランダムに割り振ることにしました。3 種類のフローは重複する部分も多かったのですが、各フローで異なる分野を詳しく掘り下げることができました。

次の 3 種類の経路に分けて実施しています。

- **AI**
- **職場環境**
- **プラットフォーム エンジニアリング**

アンケートの分析

尺度の検証

アンケートで捉えようとするコンセプトは多種多様です。表現の仕方はさまざまですが、一つの見方として、このコンセプトの尺度を変数と呼びます。これらの変数はモデルの材料であり、モデルは DORA の研究に含まれる要素です。このような尺度の妥当性を分析する方法は、大まかに内的なものと外的なものに分けられます。

尺度の内的妥当性を把握するには、あるコンセプトの存在を示すと思われるものを調べます。たとえば質の高いドキュメントがあることは、問題解決のためにドキュメントを使用する人がいることによって示される可能性があります。

調査対象の構成概念は多面的であるため、変数の大部分は複数の指標で構成されます。

変数の多面的な性質を理解するには、この概念を表すのに使用する項目がどの程度機能するかをテストします。うまく機能する場合（つまり、高い水準で同じ分散を共有している場合）、その根底には何かがある、たとえば対象のコンセプトがあると想定します。

例として、幸福について考えてみます。幸福は多面的です。幸福であれば、ある種の感情を抱いており、ある種の行動をとり、ある種の考え方をすると予想されます。特定の感情、思考、行動のパターンの根底に幸福があると想定するのです。

そのため、幸福が存在していれば、ある種の感情、思考、行動が一緒に現れると予想されます。次に、そのような感情、思考、行動についての質問をします。実際に一緒に現れるかどうかは確認的因子分析でテストします。

今年はこの分析に lavaan R パッケージを使用しました[1]。lavaan は適合に関するさまざまな統計情報を返します。これは、構成概念が質問への答えを実際に表しているかどうかを理解するために役立ちます。

あるコンセプトの指標がうまく機能しない場合は、そのコンセプトを測定する確実な方法が見つかっていないことが明らかであるため、そのコンセプトを修正するか破棄する必要が生じることがあります。

構成概念の外的妥当性とは、その構成概念がどの程度現実に適合するかを調べることです。たとえば、ある構成概念が他の構成概念と特定の関係を持つことが予想される場合があります。幸せと悲しみのように、2 つの構成概念が負の関係を持つことが予想される場合もあります。

幸せの尺度が悲しみと正の相関を示すようになれば、尺度か理論に疑問を抱くでしょう。

同様に、2 つの構成概念が正の関係を持つが強い関係ではない、ということが予想される場合もあります。生産性と仕事の満足度には正の相関がありそうですが、一般的に同一であるとは考えられていません。相関しすぎていると、同じものを測定しているようだと言われるかもしれません。これはつまり、2 つのコンセプトの違いを拾えるほどには尺度が調整されていないか、仮定した差異が実際には存在しないということです。

モデルの評価

一連の仮説に基づき、仮説モデル（物事の仕組みについて、ある側面を捉えるためのツール）を作成します。そして、収集したデータにそのモデルがどれだけ適合するかを調べます。モデルの評価では節減を追求します。これは、非常に単純化されたモデルから始めて[2]、複雑さが正当化されなくなるまで複雑さを追加するという方法です。

たとえば、組織パフォーマンスはソフトウェア デリバリー パフォーマンスと運用パフォーマンスの相互作用の産物であると予測します。単純化したモデルに相互作用は含まれていません。

```
組織パフォーマンス ~ ソフトウェア デリバリー パフォーマンス + 運用パフォーマンス
```

第 2 のモデルが相互作用を追加します。

```
組織パフォーマンス ~ ソフトウェア デリバリー パフォーマンス + 運用パフォーマンス + ソフトウェア デリバリー パフォーマンス
x 運用パフォーマンス
```

『Regression and other stories』と『Statistical Rethinking』での推奨事項に基づき[3,4]、追加の複雑さが必要かどうかを判断するために、一つ抜き交差検証（LOOCV）と渡辺・赤池の広く使える情報量規準を使用します[5,6]。

因果推論のための有向非巡回グラフ

検証済みモデルから、因果的に考えるために必要なことがわかります。ここでは、因果的に考えることの難しさについて説明します。

因果的に述べようとする理由は次のとおりです。

ここでの質問は、本質的に因果的なものだと考えられます。あることをしたらあることが起こるかどうか、ということを知ろうとしています。非因果的相関があると思われるだけの場合は、何かをするために投資しようとはしないでしょう。

分析の結果は、物事を因果的に理解しているかどうかによって異なります。回帰から得られる実際の数値は、回帰に何を含めるかによって変わります。回帰に何を含めるかは、データがどのように生成されたと考えるかによって異なるはずです（因果的主張）。そのため、明確にする必要があります。

因果的思考では好奇心が刺激され、かなりの時間がかかります。物事のさまざまな側面がどのように結びついているのか、それはなぜなのかを考えます。因果的に考えるために、物事のあらゆる側面について実験を行う必要はありません。

因果的思考は行動の中心となるものです。このレポートが行動の意思決定に役立つことを願っています。

検証済みモデルを使用することで、影響を理解するために何を考慮する必要があるかを知ることができます。簡単に言えば、データを A/B テストの形式にして、2 つの同一の状況を設定し、両者の間に 1 つだけ違いを設けます。そうすると、2 つの状況の間に生じたあらゆる違いは、最初の違いに起因するものだと考えられる、という理屈です。

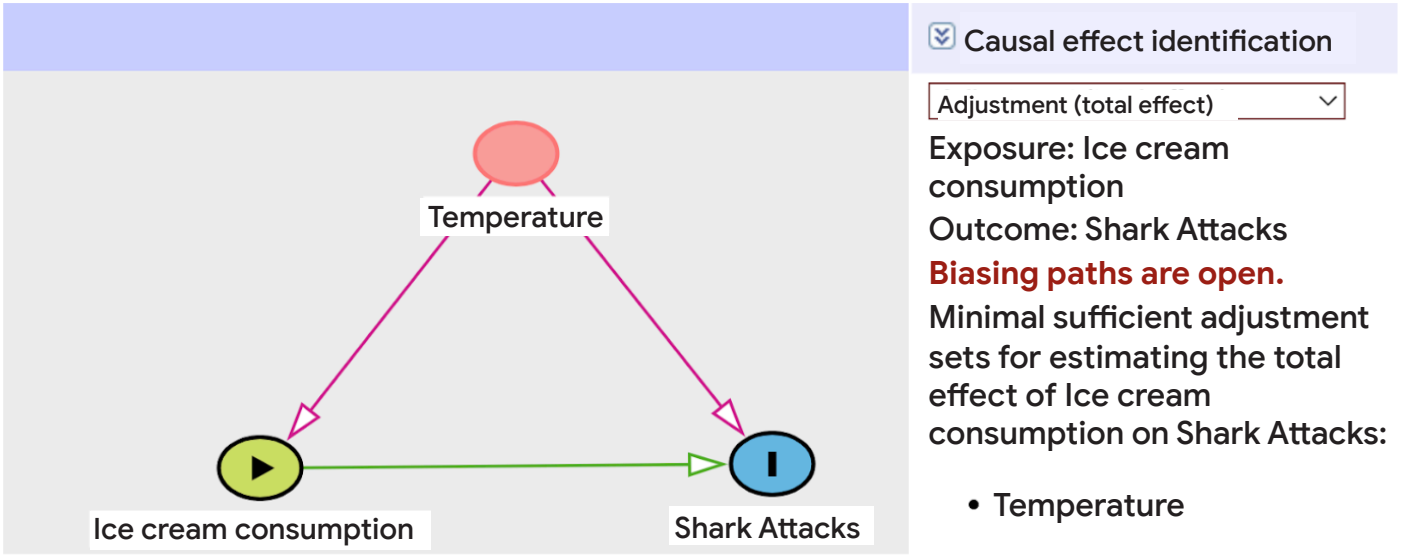
観察データやアンケート データでは、物事はそれほど明確に分けられません。参加者間では多くのことが異なっているため、交絡が生じます。DORA の因果推論の手法では、テストの模倣を試みるにあたって、これらの違いを考慮しようとします。つまり、1つのこと（AI の導入など）以外のすべてを一定に保つよう努めます。

アイスクリームがサメの襲撃を「引き起こす」という古典的な例を考えてみましょう。この観察には問題があります。つまり、暑い日に人々はアイスクリームを食べる傾向があり、また、ビーチに行く傾向もあります。アイスクリームを食べ、かつビーチに行く傾向がある、という状況と、アイスクリームを食べず、かつビーチに行かない傾向がある、という状況は、同じではありません。データがテストの論理に沿っていません。気温という交絡変数があります。

有向非巡回グラフ（DAG）は、物事がどのように異なっているのかを明らかにし、状況に対処するためのアプローチを提供します。1つの要素を除いて他のすべてを一定にすることで、テストに近い環境を再現するということです。アイスクリームとサメの襲撃の例で、アイスクリーム消費量がサメ襲撃件数に与える影響を定量化する場合、DAG がどのように役立つかを見てみましょう。

モデルを描き、理解したい効果をツールに伝えると、効果の推定にバイアスをかけるものが何なのかを教えてくれます。このケースでは、アイスクリーム消費量（Ice cream consumption）がサメ襲撃件数（Shark Attacks）に与える影響は、気温（Temperature）を調整しないと推定できない、ということが示されています。これは、アイスクリーム消費量を除くすべてを等しくし、サメ襲撃件数がアイスクリーム消費量の関数として変動し続けるかどうかを確認する、統計的アプローチです。

モデルの概要については、ご推察のとおり、モデルの章で説明しています。



画像の出典: https://www.dagitty.net/dags.html

有向非巡回グラフを使用すると、特定の効果の分析で何を考慮すべきかがわかります。

たとえば、AI の導入が生産性に与える影響を分析するときは、何を考慮する必要があるでしょうか。

ベイズ統計

この分析はベイズ統計を使用して行います。ベイズ統計にはさまざまな利点があります。

- **有意か有意でないかという考え方から脱却する（10 人に頻度論的な p 値の説明を求めれば、10 通りの答えが返ってくる）**
- **データに基づく仮説の確率を知りたいのであって、仮説に基づくデータの確率を知りたいわけではない**
- **モデルに事前知識を組み込みたい、または、少なくともどれほど知らないかを明確にしたい[7]**
- **モデリング プロセスの基本的な前提条件に対処せざるを得ない**
- **事後分布を調べることで、規模、不確実性、そして全体的にモデルがどのように、どの程度データを理解したかを知ることができる。最終的には、データから何がわかり、何がわからないのかを知ることができる**
- **非常に統一された方法で多くの統計的な問題に対処する柔軟なフレームワーク**

「シミュレーション」の意味

「シミュレーション」は、データが架空のものであるという意味ではありません。ベイズ統計学を使用して事後分布を計算し、「異なるパラメータ値が出現する頻度の予測値」を捉えます[8]。「シミュレーション」部分は、この事後分布を 1,000 回超利用し、データを基にしたとき最も信頼できるパラメータ（平均、ベータ重み、シグマ、切片など）の値を調べます。

「事後分布はパラメータ値（0.1、0.7、0.5、1 などの数字）でいっぱいのバケットのようなものです。バケットの中で、各値はその事後確率に比例して存在します。ピーク付近の値はテールの値よりもはるかに一般的です。」[9]

シミュレーションでデータの解釈の候補を調べ、不確実性の程度を把握するということです。各シミュレーションについては、データといくつかのルールしか知らない小さな AI が、情報に基づいた推測で空白（パラメータ）を埋めようとしているのだと考えることができます。これを 4,000 回繰り返すと、あるパラメータに対する小さな AI の推測が 4,000 件得られます。

こうした推測から学べることはたくさんあります。平均的な推測や、推測の 89% が該当する値区間[10]、あるレベルを超える推測の件数、推測のばらつきの程度などを知ることができます。さまざまなモデルの推測（シミュレーション）を組み合わせてみてもよいでしょう。

線がたくさんあるグラフや、可能性のある値の分布のグラフでは、データに関して最も妥当だと思われることや、不確実性の程度を示そうとしています。

調査結果をコミュニティとともに活用する

調査結果からテクノロジー主導型のチームや組織にとって貴重な視点が提供されますが、これらは、対話や共同学習を通じて理解するのが最善の方法です。DORA コミュニティでは、さまざまな情報を得て、前提を疑い、調査結果を解釈して応用する新しい方法を見つけることができます。

DORA コミュニティ（https://dora.community）に参加して、経験を共有し、他者から学び、推奨事項を実施するためのさまざまなアプローチを見つけることをおすすめします。コミュニティとともに、分析情報を活用して、組織内で有意義な変化を促進する最善の方法を探りましょう。

インタビュー

今年は、詳細な半構造化インタビューで年次アンケートを補完し、定量的な調査結果の検証、コンテキスト化、明確化を行いました。インタビュー ガイドは、アンケートに含まれるトピックに対応しており、Google Meet を介したリモート セッションを 1 回約 75 分で行うように作成されています。

プロフィールがアンケートの参加条件に一致する、合計 11 人の参加者にインタビューを行いました。インタビューはすべて動画と音声で記録しています。各セッションの所要時間は 57〜85 分であり、参加者全体で合計 14 時間 15 分のデータを収集しました。参加者のデータは「P(N)」という形式の ID で仮名化しています（N はインタビューを受けた順番に対応）。

インタビューはすべて、自動化されたソフトウェアを使用して文字起こししました。文字起こしは、アンケートのトピックをアプリオリコードとして使用し、手動でコード化しています。このレポートの最終版に掲載されている引用文は、掲載前に再確認し、手動で文字起こししました。このレポートの著者が参加者の引用文に追加した語句は角かっこ（[]）で示し、削除した語句は省略記号（..）で示しています。また、明確にするために必要な場合に限り、編集を行っています。

結果における推論上の飛躍

DORA の目標は、物事を実用的に表現したものを、仕事の進め方の改善に誰もが活用できるようにすることです。複雑なものを、単純化しています。それがこのモデルの存在理由であると言えます。ホルヘ ルイス ボルヘスの「学問の厳密さについて」という短編に、地図を 1:1 の縮尺で作る帝国の話が出てきます[11]。不条理なのは、これでは地図がまったく役に立たないということです（少なくとも私の解釈では）。DORA が行う単純化は役に立つはずです。

とはいえ、明確にしておきたい推論上の飛躍がいくつかあります。

因果関係

John Stuart Mill によると、「X が Y を引き起こす」と言うには、以下の 3 つの項目に該当している必要があります[12]。

- **相関性:** X が Y と共変動する必要がある
- **時間的な先行性:** X が Y より前に起こる必要がある
- バイアス経路が考慮されている（DAG のセクションで前述）

相関は理解できると思います。多くの場合、標準的な統計手法です。DORA のアンケートはある瞬間を捉えるため、時間的な先行性は理論的なものであり、データに含まれるわけではありません。

バイアス経路については、構造方程式モデルや有向非巡回グラフについて前述したときに触れたように、バイアス経路を考慮するための取り組みを行っていますが、これは高度に理論的な作業です。時間的先行性とは異なり、影響をデータから探ることができます。

長期間にわたる研究や正式な実験は行っていません。それでも、因果的思考こそが物事を理解する方法だと考え、因果推論の新たな手法を駆使し、最善の推定値を提供するよう務めています。相関関係は因果関係を意味するわけではありませんが、因果関係についての考え方は意味します。

ミクロレベルの現象からマクロレベルの現象へ

個人レベルの能力に着目して、それより上のレベルとどのように結びついているかを確認することはよくあります。たとえば個人の AI 導入を、アプリケーションやサービス、そしてチーム パフォーマンスに結びつけました。一見すると、あまり直感的ではありません。通常は、マクロレベルの現象が個人レベルの現象を引き起こしているというストーリーのほうが説明しやすいでしょう。「インフレーション（マクロ）が、卵を買うかどうか（ミクロ）に影響を与える」というストーリーのほうが、「卵を買わないことがインフレーションを引き起こす」というストーリーよりも受け入れやすいように思えます。

「組織のパフォーマンス（マクロ）が個人のウェルビーイング（ミクロ）に影響を与える」というストーリーも同様です。経験則からすれば、組織が個人に与える影響のほうが、個人が組織に与える影響より大きくなりそうです。

それでは、個人の行動がチームや組織のパフォーマンスに影響を与えるなどと、わざわざ言うのはなぜでしょうか。それは、完全に非論理的なわけではないと思われる、推論上の飛躍を行うからです。つまり、規模が大きくなると、以下の記述が真になる傾向があると考えています。

個人が何か（X）を行う確率は、その個人が、同じく X を行う組織やチームにいるときのほうが高くなる、と考えています。したがって、何かを行う個人は、同じく X を行う傾向のあるチームや組織を代表していることになります。もちろん、ここでのノイズはかなり大きいですが、パターンが表れるはずです。そうして、この仮定から重要なことが可能になります。

DORA 以外の例を考えてみましょう。平均身長の異なる 2 つの国があるとします。一方の国の平均身長は 5 フィート 6 インチ。他方の国の平均身長は 6 フィート 2 インチ。標準偏差は同じです。各国から無作為に 1 人ずつ選ぶとしたら、背の高いほうの人は、どちらの国から選ばれる可能性が高いと思いますか。これを何千回も繰り返せば、背の高い人が背の高い国を代表するようになるでしょう。個人の身長が、その国の身長におおむね近い値となります。

p(個人が X を行う | 組織が X を行う) > p(個人が X を行う | 組織が X を行わない)

必要なわけではありませんが、簡単なシミュレーションを行ってこれが真であることを検証しました。

```
#R code

# 再現性のために seed を設定
set.seed(10)

# 6 フィート 2 インチと 5 フィート 6 インチ
height_means = c(6 + 1/6, 5.5)

# 一定の標準偏差 1/4 フィート
std_dev =0.25

# 無作為に選ぶ
draws = 1000

# A 国から無作為に選ぶ
country_a <-rnorm(draws, mean = height_means[1], sd
= std_dev)

# B 国から無作為に選ぶ
country_b <-rnorm(draws, mean = height_means[2], sd
= std_dev)

# draws で正しい差を表す方法
represented_difference = sum(country_a > country_b)
/ 1000

# 結果をパーセント表示
represented_difference * 100
```

結果は驚くようなものではありません。無作為に 1,000 回選んだうちの 97.2% が正しい方向になります。もちろん、無作為でない選び方で、2 国間の差が小さく、サンプルが少なければ、だまされやすくなるでしょう。それでも、マクロレベルの違いはミクロレベルで表される傾向にあるという点は変わりません。

1. Rosseel Y (2012).“lavaan: An R Package for Structural Equation Modeling.”Journal of Statistical Software, 48(2), 1–36. https://doi.org/10.18637/jss.v048.i02
2. 起こり得る交絡の検討も必要となります。
3. Gelman, Andrew, Jennifer Hill, and Aki Vehtari.2021.Regression and Other Stories.N.p.: Cambridge University Press.
4. McElreath, Richard.2016.Statistical Rethinking: A Bayesian Course with Examples in R and Stan.N.p.: CRC Press/Taylor & Francis Group.
5. Gelman, Andrew, Jennifer Hill, and Aki Vehtari.2021.Regression and Other Stories.N.p.: Cambridge University Press
6. McElreath, Richard.2016.Statistical Rethinking: A Bayesian Course with Examples in R and Stan.N.p.: CRC Press/Taylor & Francis Group.
7. 事前分布が弱くなりがちであるため（懐疑的、中立、情報不足）、結果が事前分布で条件付されていないことを確認しています。
8. McElreath, Richard.Statistical rethinking: A Bayesian course with examples in R and Stan.Chapman and Hall/CRC, 2018, pg.50
9. McElreath, Richard.Statistical rethinking: A Bayesian course with examples in R and Stan.Chapman and Hall/CRC, 2018, pg.52
10. Statistical Rethinking の56 ページにある以下の McElreath の推論に従い、89% を選択しました。「なぜこの値なのか？理由はありません…この値は従来の 95% を避けています。従来の 95% 信頼区間は、無意識の仮説検定を行うよう多くの読者に促しているからです。」提示しているこの区間は、妥当な「モデルとデータに適合するパラメータ値の範囲」を示そうとしているにすぎません。
11. Borges, J. L.(1999).Collected fictions.Penguin.
12. Duckworth, Angela Lee, Eli Tsukayama, and Henry May.“Establishing causality using longitudinal hierarchical linear modeling: An illustration predicting achievement from self-control.”Social psychological and personality science 1, no. 4 (2010): 311-317.

モデル

従来、DORA では、さまざまな構造方程式モデリング手法（部分的最小二乗、共分散ベース、ベイズ）で検証した 1 つの巨大なモデルを構築してきました。2023 年のレポートでは、特定のプロセスを理解することを目的とした、多数の小規模なモデルを重視するように切り替えました。

たとえば、質の高いドキュメントの特性を理解するために、微妙な差異を反映したモデルを作成しました。特定の影響を理解するために調整された小規模なモデルを作成することには、次のような大きな利点があります[1]。

- **モデルの適合性が低い部分を簡単に特定できる。**
- **モデルに加えたものはすべて、影響を及ぼし、重みを持つ。モデルが大きくなると、変数が互いに及ぼし合うさまざまな影響をすべて理解することが、非常に困難になります。**
- **疑似相関を生むような条件付けを防ぐことができる[2]**

モデルの使い方

たくさんの疑問がありますが、多くの場合、重要な疑問は次のような形をしています。

X を行ったら、Y に何が起こる？

X は通常、手法（質の高いドキュメントの作成、AI の導入、文化への投資など）です。

Y は通常、達成や回避を目指すものです。個人レベル（生産性など）から組織レベル（マーケット シェアなど）で起こり得ます。

このような形式の疑問に対処することを目標に、モデルを作成、評価、使用しています[3]。そうして、X を行った結果として、重要な成果に何が起こるかについて、正確な推定値を提供するよう努めています[4]。影響を報告する際は、2 つの重要な特徴を伝えます。

1. 影響の**方向性**についてどの程度確信があるか。つまり、手法が有益なのか有害なのかについてどの程度明確であるか。

2. 影響の**規模**についてどの程度確信があるか。特定の手法がどの程度の影響力を持つかという相対的な意味の推定値と、推定値に関連する不確実性の度合いを提供します。

今年は次のような能力や特性を対象としました。

- **AI の導入**
- **プラットフォームの使用**
- **プラットフォームの使用年数**
- **変革型リーダーシップ**
- **優先順位の安定性**
- **ユーザー中心度**

今年の成果と成果グループは次のとおりです。

- **個人のパフォーマンスとウェルビーイング（燃え尽き症候群など）**
- **チーム パフォーマンス**
- **プロダクト パフォーマンス**
- **開発ワークフロー（コードベースの複雑さ、ドキュメントの質など）**
- **ソフトウェア デリバリー パフォーマンス**
- **組織パフォーマンス**

DORA がこうした成果に注目するのは、これら自体が最終的な目標にもなると考えているからです。もちろん、その度合いは成果によってさまざまです。組織パフォーマンスやチーム パフォーマンスがソフトウェア デリバリー パフォーマンスとは無関係だということがわかったとしたら、ソフトウェア デリバリー パフォーマンスが低くても、おそらく気にかけないでしょう。

しかし、たとえ組織パフォーマンスが個人のウェルビーイングに左右されないとしても、従業員のウェルビーイングを優先するよう考えていただきたいと思います。

繰り返されるモデル

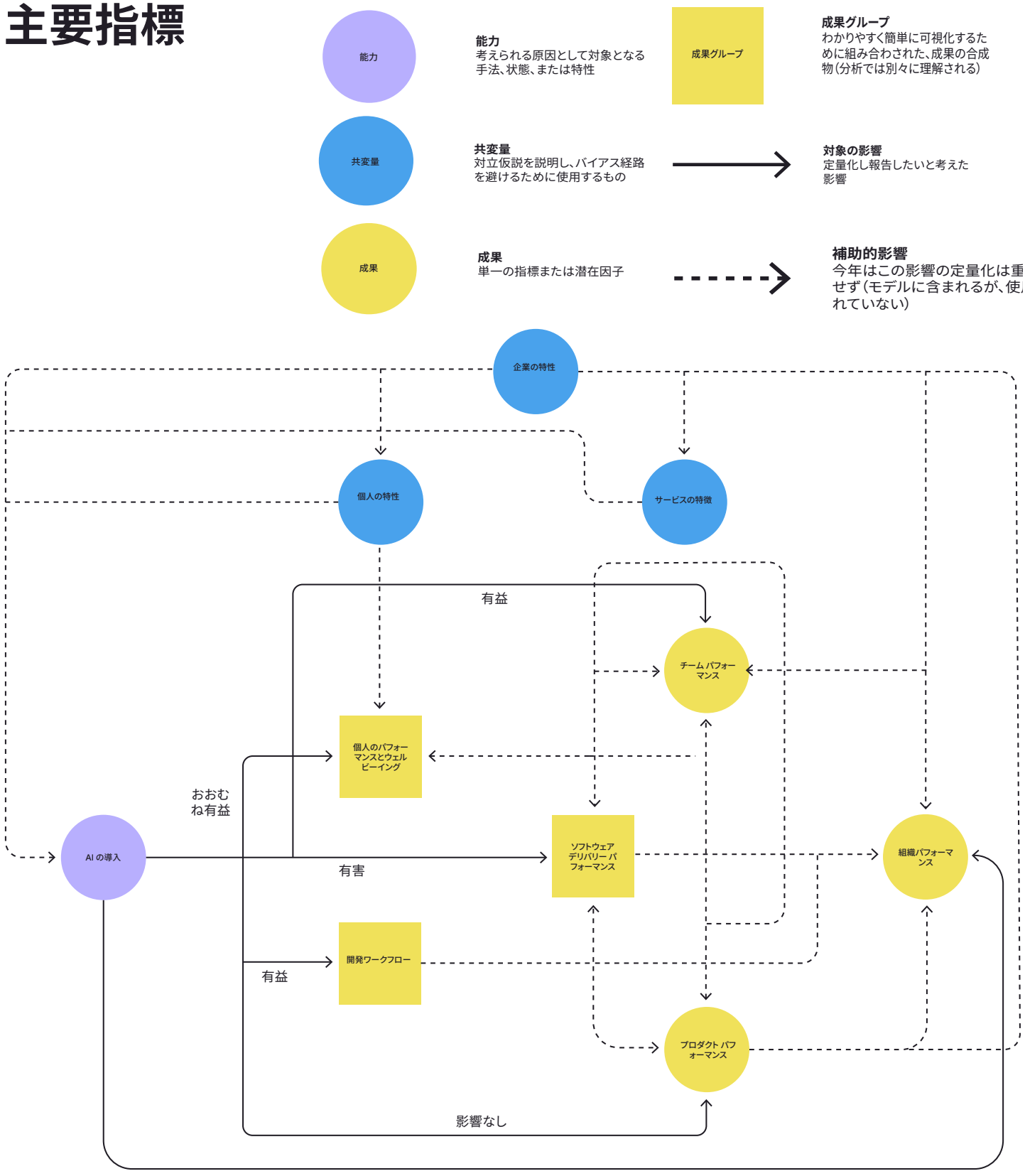
DORA は過去 3 年間、特にモデレーションとメディエーションについて、微妙な差異のある多くの仮説を立て、調査してきました。

今年はそのような仮説に注力する時間を減らし、能力が成果に与える影響を推定することに時間を割きました。つまり、各能力のモデルはほぼ同じです。

そのため、AI 導入の影響に関するモデルは、ユーザー中心の影響に関するモデルとよく似た設計になっています。モデルをコピーして能力の名前を変更することもできますが、それではあまり役に立たないでしょう。

その代わりに AI モデルを紹介します。ただしこれは各モデルの背後の図式または形式です。ご自身で分析を行う場合は、DAGitty などのツールでこのモデルを構築すれば、DORA が分析に使用した回帰をほぼ再現できるはずです。ただし、ここでは読みやすくするために少し簡略化しています。さらに、モデルは各能力で非常に似ていますが、影響は異なります。たとえば、AI の導入は一般的にソフトウェア デリバリー パフォーマンスに悪影響を及ぼすが、内部ドキュメントやユーザー中心などはその逆だということが示されています。詳しくは各章をご覧ください。

主要指標



1. Gelman らの『Regression and Other Stories』の 495〜496 ページには、参考になりそうな重要なヒントが紹介されています: B.6 多くのモデルを当てはめる、B.9 大規模な回帰の副産物としてではなく、対象を絞って因果推論を行う
2. この点に関し、『Statistical Rethinking』の第 6 章に有益な議論が記載されています。合流点バイアスについては特に参考になります。
3. これらのモデルが有向非巡回グラフとどのように結びつくのかについては、手法の章をご覧ください。
4. 因果関係については、手法の章で簡単に触れています。

関連情報

DORA コミュニティに参加して、ソフトウェアデリバリーと運用パフォーマンスの向上について話し合い、学び、協力する
https://dora.community

DORA クイック チェックを行う
https://dora.dev/quickcheck

学習環境、迅速なフロー、迅速なフィードバックを実現する能力について調べる
https://dora.dev/capabilities

生成 AI に対する開発者の信頼を醸成する
https://dora.dev/research/2024/trust-in-ai/

書籍を読む:『Accelerate: The science behind devops: Building and scaling high performing technology organizations』IT Revolution. https://itrevolution.com/product/accelerate

書籍を読む:『Team Topologies: Organizing Business and Technology Teams for Fast Flow』IT Revolution Press.
https://teamtopologies.com/

DORA の研究プログラムの出版物(過去の DORA レポートを含む) https://dora.dev/publications

研究とレポートに関するよくある質問
http://dora.dev/faq

正誤表 - 本レポートの変更点、訂正、明確化のための説明を閲覧または提案可能
https://dora.dev/publications/errata

この 2024 年の DORA レポートが最終版かどうかは以下の URL で確認できます:
https://dora.dev/vc/?v=2024.3