

THE 2026 STATE OF AI CODING REPORT

日本語版

Hanover Researchは、New Relicの委託を受けて、マネージャー職以上のテクノロジーリーダー200人を対象にアンケート調査を実施しました。これらの回答者はソフトウェアやアプリケーションの開発に生成AIまたはエージェントAIを活用している組織に勤めています。

最新の業界ベンチマークからは、ソフトウェアやアプリケーション開発におけるAI活用が進んでいることが明らかになっています。Microsoftによると、現在、同社のコードベースの30%をAIが生成しています（出典：[CNBC](#)）。また、Salesforce CEOのマーク・ベニオフ氏の推定では、ワークロードの30%~50%をAIエージェントが担っています（出典：[Bloomberg](#)）。Metaは、2026年に向けてAIが担う割合を50%にすることを目指しています（出典：[Mashable](#)）。GitHub Copilotのテレメトリーによると、ユーザー全体におけるAI生成コードの割合は平均46%です（出典：[Larridin](#)）。さらに、特筆すべきことに、Googleは最近、同社のコードのうち75%をAI生成コードが占めていると発表しました（出典：[Business Insider](#)）。

New Relicが新たに実施した調査によると、調査対象のテクノロジーリーダーの3分の2（67%）が、自社で毎週作成されるコードのうち、AIが生成または大幅なリファクタリングを担っている割合は51%~75%であると回答しています。中央値で見ると、現在、組織が運用しているコードベースの大部分が人間以外によって作成されたものです。

本レポートでは、こうした変化がもたらす影響について詳しく分析します。プラスの影響もあります。機能リリースの高速化や、デプロイ頻度の向上に加え、リーダーの大多数がAI生成コードの方が人間の書いたコードよりも品質が高いと評価しています。

プラスの効果がある一方で、マイナスの影響もあります。本番環境でのインシデント、技術的負債、マイクロサービスの乱立、上級エンジニアによる手直し時間も増加しています。

AI生成コードに起因する本番環境での障害の種類として最も多いのは、システム連携のエラー、コンプライアンス上の問題、データ完全性に関する問題、セキュリティの脆弱性の4つです。これらの障害はいずれも過去6か月間で約3割の組織に影響を及ぼしています。

最も注目すべき調査結果

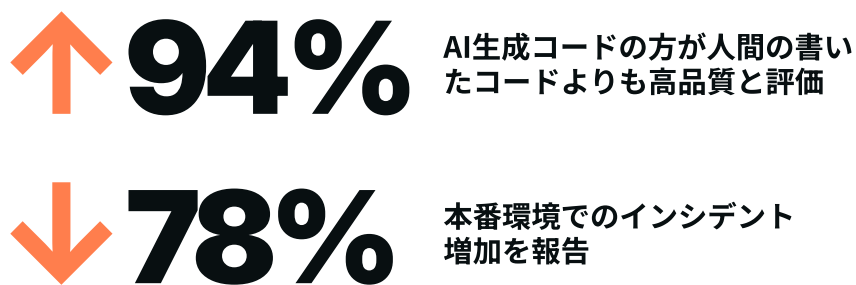
評価と現実とのギャップ

AI生成コードは、コードレビュー段階では高品質であると評価される一方、本番環境への投入後に悪影響を引き起こしていることが観測されています。本レポートではこのギャップに焦点を当てており、これは2026年のオブザーバビリティ分野における戦略的な好機となるのです。

主な調査結果

AI生成コードには、コードレビュー時の評価と本番環境での挙動との間にギャップがあります

このギャップこそが、AI活用が前提となった時代の根本的な矛盾です。レビュー段階では、94%がAI生成コードの方が人間の書いたコードよりも高品質と評価しています。しかし、いざ本番環境に投入されると、78%がインシデントが増加したと回答し、86%が上級エンジニアの緊急対応が増加したと回答しています。74%がAI生成コードの4分の1以上の大幅な修正が必要であると回答しており、さらに、82%が過去6か月間でAI生成コードに起因する本番環境での障害を1回以上経験しています。



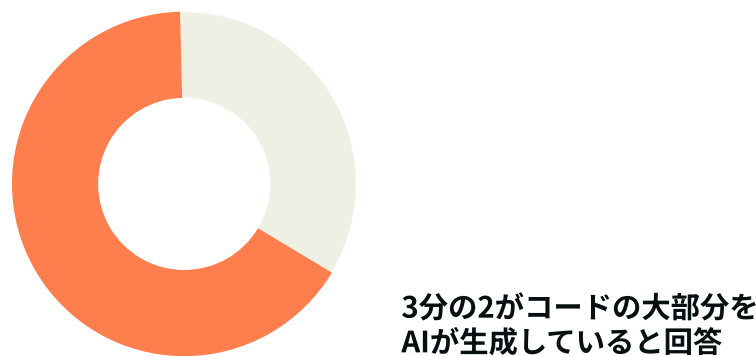
AI生成への盲信が本番環境での危機を招いている可能性がある

現代のバイブコーディングに伴う根本的な矛盾は、ライフサイクルの初期段階での過信に起因している可能性があります。テクノロジーリーダーの約3分の2（62%）が、自社のチームでは、AI生成コードへの信頼から、人手による行単位の検証を行わずにリリースすることが頻繁にあると回答しています。こうした検証を伴わない受け入れが広く行われていることが、初期レビュー段階ではコード品質が実際よりも高く評価され、いざ本番環境に投入されると問題が露呈する理由だと考えられます。



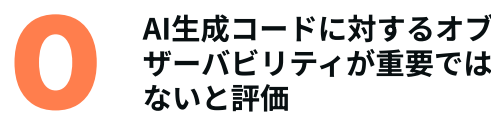
バイブコーディングは主流となりつつあります。現在では本格運用に耐えうるレベルまで進化しており、その普及速度はGOOGLEに匹敵する勢いを見せています

現在、組織の3分の2がコードの51%～75%をAI生成コードが占めていると回答しており、88%がバイブコーディングを本番環境向けの正式なルールとして明記しています。バイブコーディングを非本番環境に限定しているのはわずか5%で、全面的に禁止している組織はありませんでした。



AI生成コードに対するオブザーバビリティの確保が不可欠となり、出力にテレメトリーを含めるようプロンプトで指示する動きが広がっている

AI生成コードに対するオブザーバビリティが「非常に重要」または「極めて重要」と評価したリーダーは96%に上り、「あまり重要ではない」または「重要ではない」と評価したリーダーはいませんでした。現在、78%のチームが、生成コード自体にテレメトリーデータ（ログ、トレース、カスタムメトリック）を含めるようAIに指示しています。

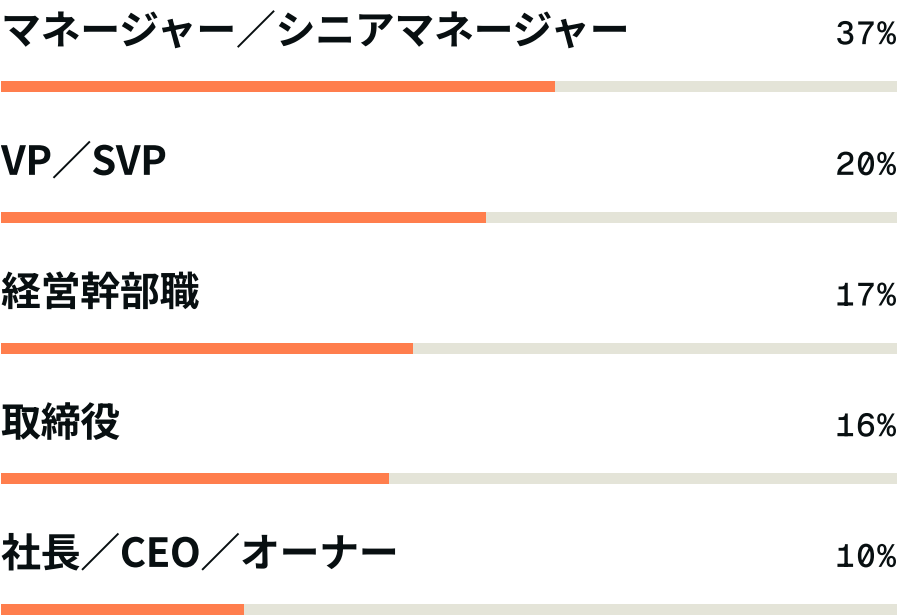


調査の方法

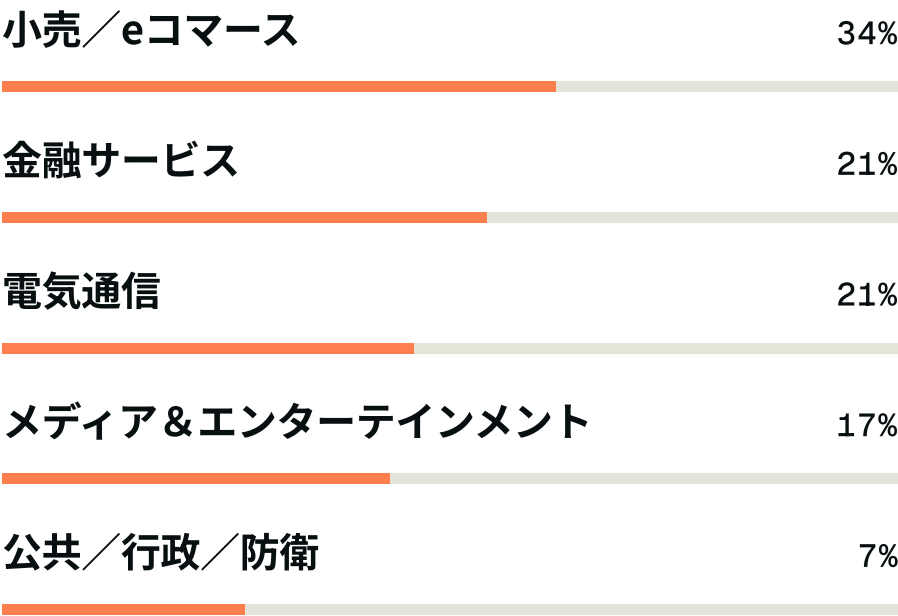
2026年、New RelicはHanover Researchに委託して、オンラインアンケート調査を実施しました。この調査は米国在住でIT・エンジニアリング分野に携わるテクノロジー意思決定者200名を対象としており、回答者は全員、ソフトウェア開発に生成AIを取り入れている組織の常勤従業員です。すべての回答者はマネージャー以上の職に就いており、ソフトウェアの購買に関する実質的な権限を持っています（主たる意思決定者：65%、共同意思決定者：34%）。

本レポートの数値は、回答者200名を母数とする割合で示されます。本文中に言及している主要な統計値は、分布の中で最も注目すべき項目を取り上げています。一方で、グラフでは各データポイントの背景にある分布全体を示しています。

回答者の職位



業界区分



組織の売上規模が中堅上位企業・大企業層に集中



ニューノーマル

AI生成コードの活用は、サンドボックス環境での実験段階を終え、ソフトウェア開発ライフサイクルに正式に組み込まれています。

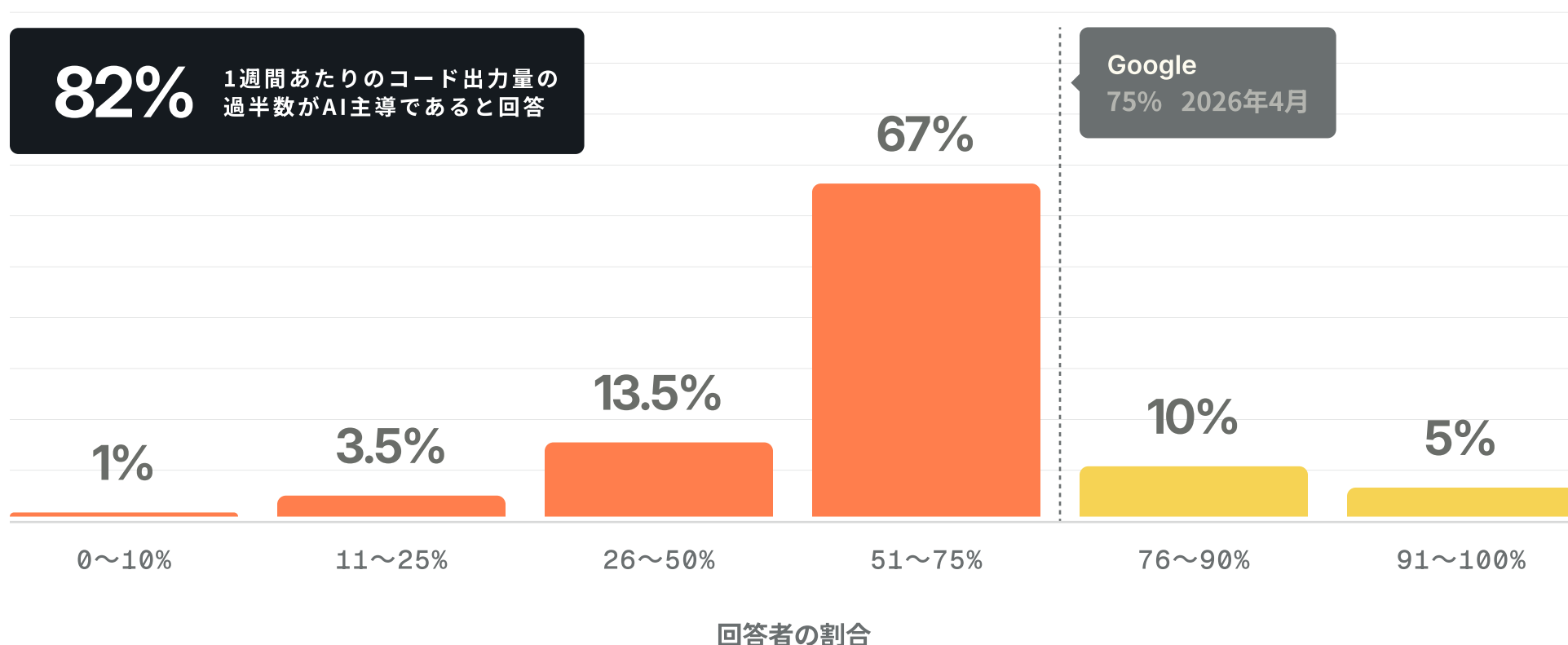


コーディングにおけるAI活用は 今や標準となっている

リーダーの67%が、自社における1週間あたりのコード出力量の51%～75%をAI生成コードが占めていると回答しています。

1週間あたりの出力量のうち、AIが生成または大幅にリファクタリングしたコードの割合

N=200



この結果が示すこと

Googleが発表した「75%」という数字は、当初は外れ値として広く捉えられていました。しかし、本調査のデータはその認識を改めるものです。現在、米国のテクノロジーリーダーの間では、51%～75%の範囲が最も一般的な回答となっています。ほとんどの回答者の組織では、AIが主にコード作成を担い、人間はそのレビューを担当しています。

こうした割合を考慮すると、本番環境でのインシデントに対応するエンジニアがその原因となったコードを自ら書いている可能性は統計的に低いと言えます。その結果、「システムを理解する」ことの意味が変化しています。

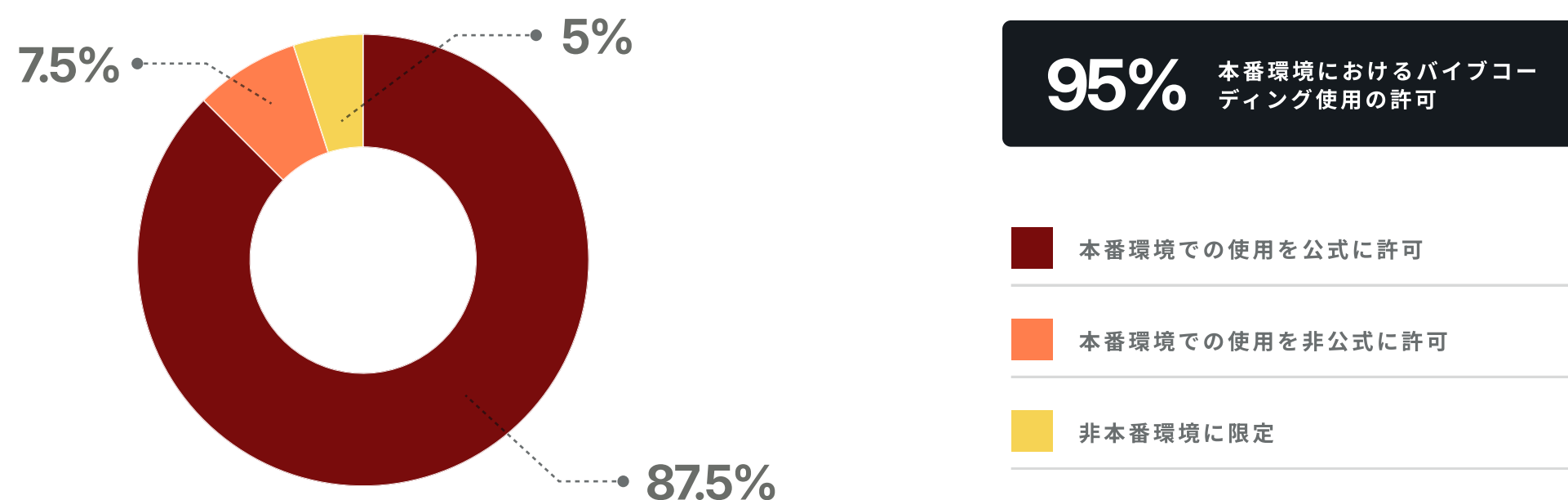
ソースコードを読むことは、もはやシステムを理解するための第一の手段ではありません。生成されるコード量はあまりにも膨大で、生成速度も速く、多様なプロンプトが用いられています。そのため、実行証跡（コードが実際に何を行い、どの入力を受け取り、どの順序で実行され、下流にどのような影響を及ぼしたのかを示す情報）が、システムを理解するうえで、より信頼できるシグナルとなります。「実際に起こっている事象」を把握するための信頼できる情報源としてオブザーバビリティプラットフォームを活用しているエンジニアリング組織は、今もなおシステム理解をコードレビューに依存している組織よりも、よりスムーズなスケーリングが可能です。

バイブコーディングが本番環境の 正式なルールに組み込まれる段階へ

バイブコーディングを非本番環境に限定している組織はわずか5%で、全面的に禁止している組織はありませんでした。残りの95%は、本番サービスでの使用を公式（87.5%）または非公式（7.5%）に認めています。

バイブコーディングに対する組織の方針

N=200



この結果が示すこと

バイブコーディングは、副次的な取り組みでも、個人の生産性を高める工夫でも、サンドボックス環境で用いるテクニックでもありません。調査対象となった組織の多くでは、バイブコーディングがソフトウェア開発サイクル（SDLC）に組み込まれています。AI生成コードは上級エンジニアが書いたコードと同じ本番環境、収益を生むサービス、顧客向けエンドポイントに投入され、同じSLAと障害対応時の時間的制約の下で運用されています。

バイブコーディングが本番環境ポリシーに組み込まれると、コード作成者にかかわらず、以下のような関連するガバナンスシステムをすべて等しく適用することが求められます。

コードレビュー、変更管理、デプロイ時のガードレール、インシデント対応、監査ログ。

本レポートの後半で紹介するデータにより、ほとんどの組織が関連する管理体制を十分に強化しないまま、バイブコーディングの実践を承認している現状が浮き彫りになっています。

人間が書いたコードに適用しているリリース健全性、変更追跡、SLO監視といった既存の規律をAI生成コードにも適用し、このギャップを早期に解消できれば、バイブコーディングされたサービスを別物として扱う場合よりも、運用に与える衝撃をスムーズに吸収できるでしょう。

AI生成コードは人間が書いたコードよりも高く評価されている

61%のリーダーが、AI生成コードは人間の書いたコードよりも「やや品質が高い」と評価しています。さらに、33%が「大幅に品質が高い」と評価しています。

AI生成コードと人間が書いたコードに対する品質評価の比較



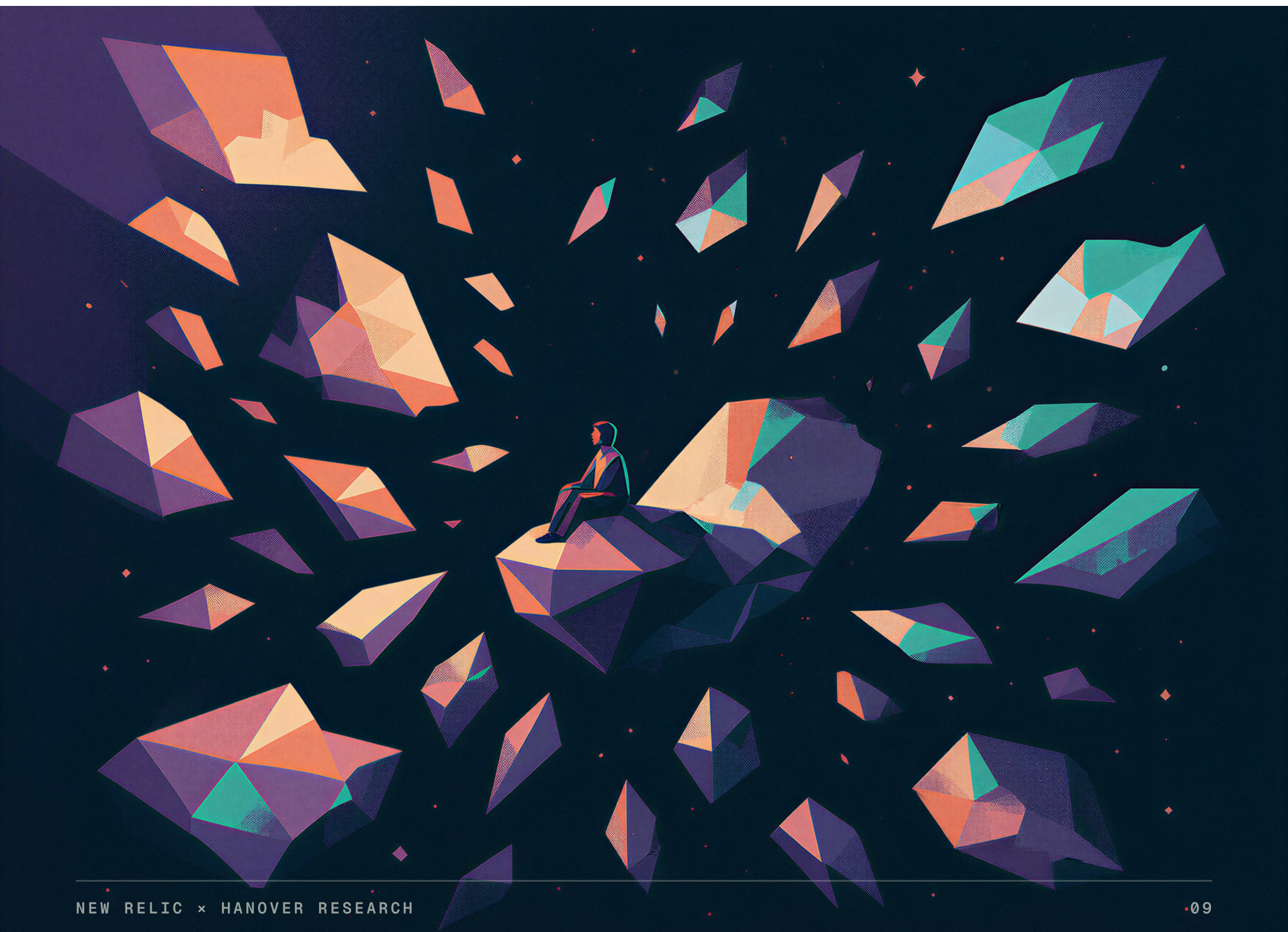
この結果が示すこと

AIによるコーディングの最大のメリットは、大半のエンジニアリング組織が品質を評価するコードレビューの段階で、期待どおりの成果を上げていることです。AI生成コードには、コード提出段階において、整理された書き方、一貫したスタイル、明らかなバグの少なさといった強みがあります。こうした認識は現実在即したものであり、現場で広く浸透しています。組織がバンプコーディングの導入をさらに進めようとしている理由の1つでもあります。

ただし、注意すべき重要な点は、この品質評価がインシデント対応段階ではなく、コードレビュー段階での判断であるということです。コードが読みやすいことと、それが正常に動作するかどうかは別の問題であり、それぞれを測定する指標も大きく異なります。本調査のデータからは、高く評価されたコードが実際のトラフィック下、依存関係内、顧客がいる環境で稼働し始めたときに何が起こるのかが見えてきます。

実運用の 現状

AI生成コードによって出力速度は向上したものの、システムの信頼性を維持するためのコストと作業負荷は増大しています。



運用負荷も並行して増加

過去12か月間で、上級エンジニアが手直しに費やす時間が増加した組織は86%に上ります。また同期間において、マイクロサービスの乱立、本番環境でのインシデント、技術的負債のいずれもが、4分の3以上の組織で増加しています。

大幅な手直しを要するAI生成コードの割合

N = 200

環境内のマイクロサービス／関数の数



上級エンジニアがコード修正に費やす時間



本番環境でのインシデント発生数



技術的負債



減少した

増加した

この結果が示すこと

上記の4つの指標は、AIを活用した開発に伴う典型的な「下流コスト」であり、開発速度や評価上の品質が向上した12か月間に連動して増加しています。開発速度の向上は低コストで実現可能になりました。一方、信頼性の確保はそうではありません。開発速度の向上を報告している組織は、同時に、ここに示した運用負荷も引き受けています。

これはAI活用に伴う主要なトレードオフであり、このことはデータによって裏付けられています。このようなトレードオフへの適切な対応戦略は、開発速度を落とすことではありません。開発速度の向上は事実であり、その効果は収益にも表れています。

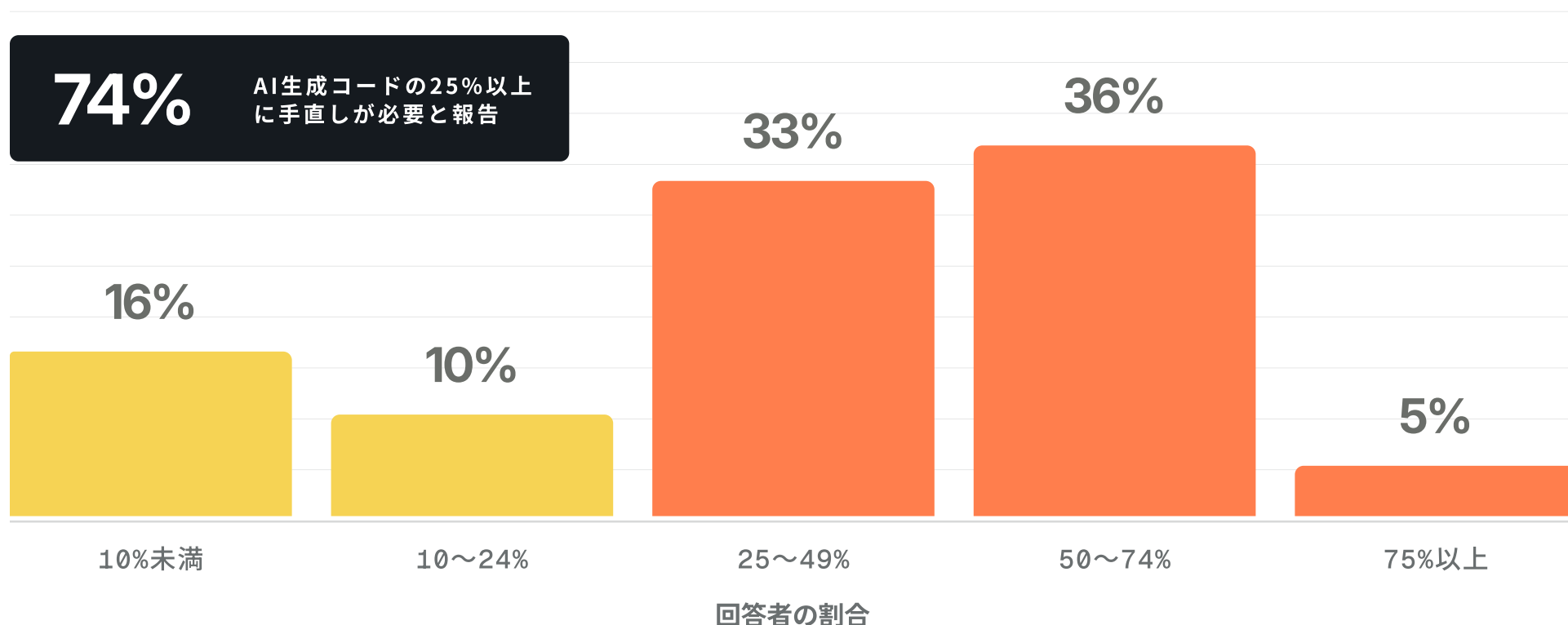
機能のリリース速度向上を背景に、63%が収益の緩やかな増加を報告しています。適切な対応策は、開発速度の向上に伴う運用負荷をあらかじめコストとして明確に想定しておくことです。つまり、デプロイ頻度や機能スループットを対象とする同じキャパシティ計画において、インシデント発生率、MTTR、技術的負債への対応、上級エンジニアの手直し時間も併せて考慮する必要があります。開発速度向上の効果とそれに伴う運用負荷の両面を経営陣に提示できれば、どこで開発を加速させるべきか、どこで時間を設けて補強すべきかについて、よりの確な判断を下せるようになります。

AI生成コードによって手直しが増加

41%のリーダーは、AI生成コードの少なくとも半分が、コンテキストの不足、データの不備、前提の誤りに起因する大幅な手直しが必要になると回答しています。また、74%は、少なくとも4分の1のAI生成コードが同様の手直しが必要になると答えています。

大幅な手直しを要するAI生成コードの割合

N=200



この結果が示すこと

バンプコーディングには、後工程での多大な修正コストが伴います。コードの大半をAIに生成させている組織であっても、その出力の無視できない割合が修正のために差し戻されています。これは仕上げを目的としたものではなく、コンテキストの不足、架空のインターフェースを前提とした記述、データに関する前提の誤りなどが原因です。本来はより重要度の高い課題に取り組めるはずの上級エンジニアが、このような手直し作業を担うケースが増えています。

この根本原因は、AIコーディングツールが実行時におけるシステムの実際の挙動を把握しないままコードを生成していることです。

AIコーディングツールはソースコードを読むことはできても、トレースを読み取ることはできません。本番環境のテレメトリ、実際のクエリパターン、実際の依存関係グラフを開発プロセスにフィードバックすることで、このギャップの解消に着手した組織では、手戻りの発生率が最も低くなっています。実行時データとコード生成との間にフィードバックループを構築することは、今後24か月間で最も重要なオブザーバビリティの取り組みとなる可能性があります。これが運用負荷の軽減に最も直結する対応でもあります。

組織に影響を及ぼす障害の種類

過去6か月間に、AI生成コードに伴う問題がまったく発生しなかったと報告した組織はわずか19%でした。一方、82%は本番環境での問題を平均で2件以上挙げています。

過去6か月間に発生したAI生成コードに起因する本番環境での主な問題

N = 200



この結果が示すこと

4つの障害の種類（システム連携のエラー：31%、コンプライアンス／ガバナンス上の問題：30%、データ完全性に関する問題：29%、新たなセキュリティ脆弱性の混入：28%）はいずれも、過去6か月間で約3割の組織に影響を及ぼしました。従来、人間が書き、ピアレビューを経たコードでは障害発生率が一定のベースラインにとどまっていたましたが、AI生成コードでは実行時に深刻な問題が発生する割合が1.7倍高くなっています。AI生成コードが本番環境で障害を引き起こすパターンは、1つに限られません。AI生成コードは多数の小さな問題を同時に発生させ、本番環境で障害を引き起こします。その影響は広範囲に及ぶため、ほとんどの組織が過去6か月間に1件以上の障害対応を経験しています。

ここに挙げた4つの障害モードには、それぞれ異なる観測可能な兆候が表れます。

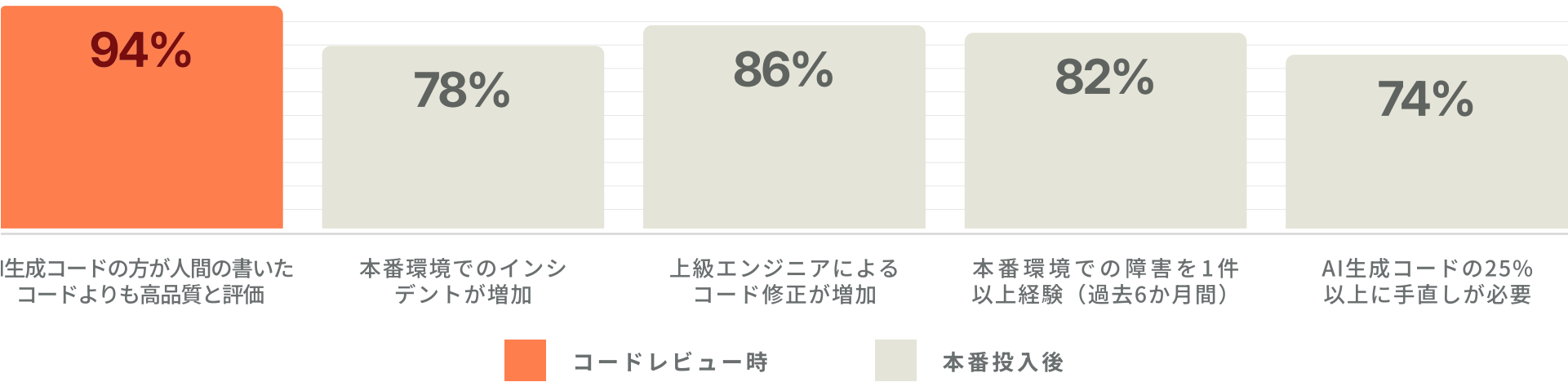
統合の失敗は、スキーマのドリフト、コントラクト違反、サービス間のエラー率上昇といった兆候として表れます。コンプライアンス上の問題は、監査ログやポリシー違反といった形で表れます。データ完全性の問題は、本番環境のテレメトリー上で、重複、ドリフト、想定外のnull率といった異常として表れます。セキュリティの脆弱性は、トレースデータ、認証パターン、異常な挙動といった形で表れます。各障害モードをオブザーバビリティプラットフォーム上の検知パターンと紐付けて監視することで、AIによるミスが本番環境に到達しても、実害が出る前に検知することができます。そのような組織は、依然としてユーザーからの苦情や定期監査に依存している組織よりも、検知までのギャップをはるかに速く埋めることができます。このような事情を背景に、一元的なテレメトリー基盤が現場で求められるようになっており、New Relicや同様のプラットフォームでAI生成コードに関するワークロードが増加している一因にもなっています。

AI支援時代の中心にある矛盾

レビュー段階ではAI生成コードの方が人間の書いたコードよりも品質が高いと評価していた組織（94%）が、そのコードを本番環境に投入した後は、インシデントの増加（78%）、手直しの増加（74%がコードの25%以上に手直しが必要と回答）、上級エンジニアによる緊急対応の増加（86%）、本番環境での障害の増加（82%が過去6か月間で障害を経験したと回答）を報告しています。

レビュー段階では優れて見えるが、本番環境では問題が生じやすいAI生成コード

N=200



この結果が示すこと

これは本調査で最も重要な傾向です。レビュー担当者の目には、人間が普段書くコードよりも整理され、一貫性があり、適切に構成されたコードとして映ります。一方、本番環境では、より頻繁に、より多様な形で障害を引き起こすコードとなり、修正には上級エンジニアによる対応が多く必要になります。どちらの見方も正確です。ただし、両者が見ているものは同じではありません。前者が評価しているのは、読みやすさと表面的な品質です。一方、後者が評価しているのは、実際の負荷、実際の依存関係、実際のエッジケースにおける挙動であり、AIコーディングツールはコード生成時にこうした状況を把握することはできません。

この矛盾を解消する鍵は1つです。レビュー段階では優れて見えるAI生成コードが、本番環境では問題を引き起こし

やすい理由は、AIが人間のレビュー担当者にも見える情報（ソース）をもとに作業している一方で、本番環境でしか生成されない情報（トレース）を認識できないためです。

このようなギャップを解消することが、今後数年間にオブザーバビリティレイヤーが担うべき戦略上の役割です。AI支援時代に成功を収める組織とは、本番環境のテレメトリを、単なる事後的なデバッグツールではなく、開発プロセスにおける主要なインプットとして扱う組織です。つまり、本番環境のテレメトリを次のプロンプト、次のコードレビュー、次のデプロイ判断を改善するためのフィードバック信号として活用できるかどうかが鍵となります。これこそがオブザーバビリティという分野が担うべき役割であり、本調査のデータは、その重要性を何よりも明確に示しています。

AI生成への盲信が本番環境での危機を招いている

62%のテクノロジーリーダーが、自社のエンジニアリングチームでは、AI生成コードへの信頼から、人手による行単位の検証を行わずに本番投入することが「頻繁に」と回答しています。

行ごとの検証をせずにAIが生成したコードを信頼する回答者

N=200

62%

頻繁に／常に

この結果が示すこと

現代のバンプコーディングに伴う根本的な矛盾は、開発ライフサイクルの初期段階での過信に起因している可能性があります。こうした受け入れが広く行われていることが、初期レビュー段階ではコード品質が実際よりも高く評価され、いざ本番環境に投入されるとすぐに問題が露呈する理由だと考えられます。

単体の最適な環境であれば、LLMは完璧に動作するコードを生成する能力に優れています。一方で、エッジケース、並行処理の死角、古いAPIの非推奨化、複雑な状態変化などの要素を見落としがちです。

十分な検証を経ずにリリースされた場合、このような構造上の見落としは、実際のユーザートラフィックにさらされるまで表面化せずに潜み続けます。

AI生成コードを検証なしに本番投入すると、本番環境での局所的なインシデントが大幅に増加する可能性があります。現在、SREやDevOpsチームが「AIコードの後始末」を強いられるケースが増えており、本来はロードマップの加速に充てられるはずの稼働時間の最大3分の1を、未検証の機械出力のトリアージやリファクタリングに費やしている状況です。

今後の 戦略

テクノロジーリーダーは、AI生成コードを管理するうえでオブザーバビリティが不可欠であると声を揃えています。

オブザーバビリティは今や不可欠

39%のテクノロジーリーダーが、AI生成コードを活用するうえでオブザーバビリティが「極めて重要」と評価しており、これは単体で最も選ばれた回答でした。さらに57%が「非常に重要」と評価しています。「あまり重要ではない」または「重要ではない」と評価した回答者はいませんでした。

AI生成コードを活用するうえでのオブザーバビリティの重要性

N=200



この結果が示すこと

AI生成コードにオブザーバビリティが不可欠かどうかをめぐる議論は事実上決着を見せています。「あまり重要ではない」または「重要ではない」と評価した回答者がいなかったことを踏まえると、焦点となっているのは、もはやオブザーバビリティへの投資の必要性ではなく、どのプラットフォームを標準とすべきかです。重要性評価がこれほど上位に集中している状況では、戦略的な取り組みの焦点が購買から統合へと移っていると言えます。調査対象の組織の大半はすでにオブザーバビリティスタックを運用しており、複数のツールを併用しているケースも多く見られます。

ほぼ全員に共有されているこの共通認識をもとに最大限の効果を引き出すリーダーとは、オブザーバビリティとAIOpsを幅広くカバーする単一のプラットフォームへの標準化を進める人たちです。深夜2時にオンコール対応しているエンジニアの誰からも信頼されていないような、分断された監視カバレッジに対してコストを払い続けることはありません。

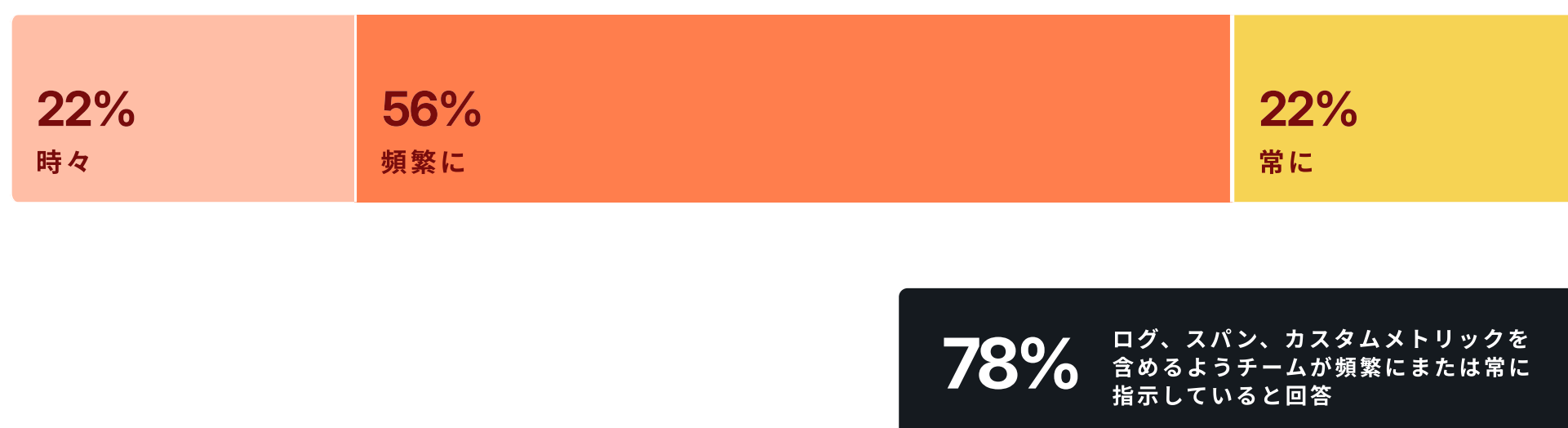
4つのシグナルすべてを単一のデータモデルとクエリサーフェスの下に統合するNew Relicのようなプラットフォームは、このような統合の機会を捉えるうえで有利な立場にあります。

AIプロンプトへの組み込みが進む テレメトリー

56%のリーダーが、自社のチームでは、コード生成時に特定のテレメトリー（ログ、トレース、カスタムメトリック）を含めるようAIに指示することが「頻繁に」と回答しています。さらに22%は「常に」指示していると回答しています。「稀に」または「まったくない」と答えたリーダーはいませんでした。

生成コードにテレメトリーを含めるようチームがAIに指示する頻度

N=200



この結果が示すこと

エンジニアはコードを書き終えてからオブザーバビリティを考えるわけではありません。現在のエンジニアはオブザーバビリティの考慮を前倒しし、プロンプト自体に組み込むようになっていきます。テレメトリーは、AIによるコード生成の「完了」条件の1つとなりつつあります。

ログ記録の対象、トレースの対象、アラートの対象の選定は、SREのバックログから開発者のプロンプトへと移り、より上流の工程で行われるようになっていきます。このことから、2つの示唆が導かれます。第一に、組織のテレメトリーの品質がエンジニアの用いるプロンプトやテンプレートに大きく左右されるようになっていきます。

第二に、方針が明確でスキーマを意識した規約（OpenTelemetryのセマンティック規約、構造化ログ、一貫性のあるスパン属性）を標準化する組織は、開発者ごとに独自の形式でプロンプトを入力させる組織よりも、シグナル対ノイズ比が大幅に向上すると考えられます。Cursor、Copilot、Claude Code、ChatGPT上で使われる標準を定める組織が、事実上、このカテゴリの次の段階におけるオブザーバビリティの標準を決定することになります。

まとめ

3つの調査結果により、エンジニアリング組織が2026年の残りの期間で取り組むべき方向性が示されます。

開発速度の向上が現実となった一方で、その代償は実行時に生じます

リリースの高速化、デプロイ頻度の向上、レビュー時の品質向上はいずれも、期待どおりに実現しています。そのためのコストは、インシデント、手直し、マイクロサービスの乱立、上級エンジニアの対応時間といった形で発生しています。その両面を見据えて計画を立てることで、リーダーはどこを加速させ、どこを補強すべきかについて適切に判断できます。

本調査で対応に最も直結する知見は「懸念」と「管理策」の間にあるギャップです

リーダーはAI生成コードに伴うリスクを正しく認識しています。一方で、実際に導入されている管理策は、その懸念に見合う水準には達していません。セキュリティスキャン、デプロイ時のガードレール、ガバナンスポリシー、実行時のオブザーバビリティなど、多層にわたる安全措置を用いてこのギャップを最速で埋める組織こそが、AI生成コードが多くを占めるコードベースを、許容範囲を超えるリスクを負うことなく、エンタープライズレベルのSLAの下で運用できるようになります。

レビュー時の品質と本番環境での現実との間にある矛盾こそが、オブザーバビリティの真価が発揮される領域です

AIコーディングツールはソースコードを認識します。一方で、トレースを読み取ることはできません。フィードバックループを構築しない限り、AI生成コードはレビュー段階では高い評価を受けながら、本番環境ではチームに予期せぬ問題をもたらし続けるでしょう。オブザーバビリティプラットフォームは、AI生成コードの実際の動作を可視化するための信頼できる情報源となり、そのシグナルを開発プロセスにフィードバックする役割を担っており、AI支援時代のエンジニアリング組織の運用を支える中核的な存在となります。New Relicはまさにこのフィードバックループを軸に設計されており、真っ先に投資すべきレイヤーであると言えます。

New Relicがこの調査結果を発表した理由は、この調査の内容が、すべてのエンジニアリング組織が現在取り組んでいる課題に答えるものだからです。この調査のデータからは、AIがソフトウェアを生成する時代が到来し、その運用による影響が顕在化しており、オブザーバビリティがそのような影響に対処するレイヤーとなっていることが明らかになっています。

HANOVERについて

Hanover Researchは、市場調査・分析を手掛けるグローバル企業であり、定額制のサブスクリプションモデルを通じて定量的・定性的なカスタムインサイトを提供しています。同社は2003年に創業し、現在では200名を超えるアナリスト、統計専門家、調査手法の専門家を擁しています。Hanover Researchは、企業、医療、教育の各領域における戦略上、事業上、財務上の意思決定を支援するため、一次市場調査、予測データモデリング、セグメント分析を行っています。

NEW RELICについて

New Relicインテリジェントオブザーバビリティプラットフォームは、ソフトウェアスタック全体にわたるテレメトリデータを取り込み、標準化し、分析します。New Relicは、メトリクス、イベント、ログ、トレースを単一のテレメトリデータプラットフォームに集約することで、システムエンジニアリングを事後的なデバッグから脱却させ、予測に基づくパフォーマンス管理へと移行させます。そのため、New Relicは、インフラストラクチャの健全性監視やアプリケーションパフォーマンスの最適化のソリューションとして、Adidas Runtastic、米国赤十字社、ドミノ・ピザ、GoTo Group、Ryanair、Topgolf、William Hillといった組織から選ばれています。

本レポートについて

本レポートのデータは、New Relicの委託を受けてHanover Researchが2026年に実施したオンラインアンケート調査に基づいています。この調査対象は、ソフトウェア開発に生成AIを活用している組織に勤務し、米国在住でIT・エンジニアリング職に就いている常勤のテクノロジー意思決定者200名です。

New Relicのプラットフォームについて知る