



生成 AI AWS を使用して でソフトウェア開発ライフサイクルを加速する

AWS 規範ガイドンス



AWS 規範ガイド: 生成 AI AWS を使用して でソフトウェア開発ライフサイクルを加速する

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

序章	1
目的	1
対象者	2
開発経験	3
生成 AI の使用	4
5-I フレームワーク	5
フレームワークの概要	5
SDLC との統合	7
基本的な機能	8
プロジェクト管理	14
要件管理	17
アーキテクチャと設計	18
コラボレーション	18
DevSecOps	19
オペレーションとメンテナンス	27
AI アシスタント	29
分析とインサイト	31
ナレッジ管理	33
拡張性	34
ベストプラクティス	36
統合されたツールチェーン	36
DevSecOps パイプライン	37
コラボレーションツールとプラクティス	37
タスクの自動化	37
レビューと反復	38
プロジェクト管理プラクティス	38
ナレッジ管理	39
拡張性とカスタマイズ	39
最適化	39
データ駆動型インサイト	40
プラットフォームベースのアプローチ	40
成功の測定	41
デプロイ速度	41
コード品質	42

運用効率	42
チームの生産性と満足度	43
ビジネスへの影響	43
結論	45
リソース	45
ドキュメント履歴	47
.....	xlviii

生成 AI AWS を使用して でソフトウェア開発ライフサイクルを加速する

チェタン・マクバナ、アマゾン ウェブ サービス

2025 年 4 月 ([ドキュメント履歴](#))

高品質のソフトウェアに対する需要の高まりにより、組織はソフトウェア開発ライフサイクル (SDLC) を加速する方法を常に模索しています。組織が競争力を維持するには、製品品質を維持または改善しながら、市場投入までの時間を短縮することが重要です。これらの課題に対処するには、ソフトウェア開発エクスペリエンスを進化させ、プロセスを合理化し、ソフトウェア開発チームが生産性と創造性を高める最先端のテクノロジー、方法論、プラクティスを使用する必要があります。次世代の開発エクスペリエンスの出現は、ソフトウェアの構想、構築、テスト、デプロイ方法の大きな変化を示しています。クラウドネイティブ開発、AI 駆動型オートメーション、高度なプロジェクト管理、コラボレーションツール、DevSecOps など、さまざまな機能が統合されており、SDLC の効率と有効性をまとめて強化します。

この変換の最前線にあるのは、ソフトウェアエンジニアリングにおける生成 AI の台頭です。[ガートナー](#)によると、2023 年の 5% と比較して、プラットフォームエンジニアリングチームの 40% が AI を使用して 2027 年までに SDLC のすべてのフェーズを強化します。このレポートでは、ソフトウェアエンジニアリングのリーダーは、開発プロセスにとって重要な幅広い分野で生成 AI を採用する準備をする必要があることも述べています。別のレポートでは、[McKinsey](#) の調査によると、デベロッパー速度インデックスが高い企業は収益が 4~5 倍速く、株主収益が 60% 高く、イノベーションが 55% 高いことがわかります。コード生成だけでなく生成 AI を採用することで、組織はソフトウェア開発ワークフローで新しいレベルの効率、生産性、イノベーションを引き出すことができます。これにより、手動作業を減らし、インサイトを明らかにし、人間の専門知識を高めることができます。

目的

この戦略ドキュメントでは、生成 AI を使用して SDLC を高速化するのに役立つフレームワーク、基本的な機能、ユースケース、ベストプラクティス、成功メトリクスの概要を説明します。製品の品質と効率を向上させるために、すべての開発段階で生成 AI を効果的に統合する方法について説明します。

この戦略ドキュメントは、お客様とお客様の組織が次の目標を達成するのに役立ちます。

- フレームワーク、基本的な機能、ユースケース、ベストプラクティス、成功メトリクスを実装して、生成 AI で SDLC を高速化します。

- すべての開発段階で生成 AI を効果的に統合し、製品の品質、リリース速度、開発効率を向上させます。
- プロセスを合理化し、開発チームを強化する最先端の AI テクノロジー、方法論、プラクティスを組み込むことで、次世代のソフトウェア開発に適応します。

対象者

この戦略ドキュメントは、IT リーダー、エンジニアリングマネージャー、最高技術責任者、ソフトウェア開発チームが、開発プラクティスに生成 AI を適用してソフトウェア開発のライフサイクルを加速したいと考えている方を対象としています。

ソフトウェア開発エクスペリエンスについて

ソフトウェア開発エクスペリエンスには、ソフトウェア開発ライフサイクル (SDLC) 全体で開発チームが使用する環境、ツール、プロセスが含まれます。これには、統合開発環境 (IDE)、コラボレーションプラットフォーム、テストフレームワーク、ナレッジ管理システム、デプロイパイプラインなどが含まれます。

適切に設計された開発エクスペリエンスにより、ワークフローが合理化され、手動作業が軽減され、チームが価値の高いタスクに集中できるようになり、最終的に SDLC が高速化されます。例えば、IDE、バージョン管理システム、デプロイツールをシームレスに統合することで、デベロッパーは、手動のハンドオフとコンテキスト切り替えを必要とするフラグメント化されたツールチェーンと比較して、コードの書き込み、テスト、デプロイの速度と効率が向上します。同様に、堅牢なナレッジ管理フレームワークを統合することで、チームは組織の知識、ベストプラクティス、ドキュメントに簡単にアクセスして共有できます。これにより、全体的な生産性と問題解決能力が向上します。

ソフトウェア開発の経験は、ソフトウェア開発チームの全体的なパフォーマンスと成功に直接影響します。最適ではないエクスペリエンスは、以下につながる可能性があります。

- 生産性の低下 – 非効率的なツール、複雑なワークフロー、自動化の欠如によりチームの生産性が低下し、機能や更新の提供が遅くなります。
- 技術的負債の増加 – ツールとアドホックプロセスの統合が不十分な場合、技術的負債が発生する可能性があります。時間の経過とともにソフトウェアシステムの保守とスケーリングがより困難になります。
- イノベーションの低下 – 手動の反復的なタスクに悩まされると、チームが新しいテクノロジーを探索し、イノベーションを推進する能力が制限されます。
- 侵害された品質 – 断片化されたテストとデプロイプロセスにより、ソフトウェアの欠陥や脆弱性のリスクが高まります。これは、配信されるソフトウェアの全体的な品質に悪影響を及ぼす可能性があります。

適切に設計されたソフトウェア開発エクスペリエンスに投資することで、市場投入までの時間の短縮、ソフトウェア品質の向上、ソフトウェア開発チームの満足度の向上、ビジネスの俊敏性の向上など、大きなメリットを引き出すことができます。

生成 AI によるソフトウェア開発エクスペリエンスの強化

生成 AI をソフトウェア開発ライフサイクル (SDLC) に統合することは、ソフトウェア開発チーム全体のソフトウェアソリューションの構想、設計、実装、保守方法のパラダイムシフトを表します。生成 AI は、プロジェクト管理、要件収集、設計、コーディング、テスト、デプロイ、メンテナンスなど、SDLC のあらゆるフェーズに変革をもたらす可能性があります。

生成 AI を活用した開発エクスペリエンスは、製品マネージャー、デザイナー、ソリューションアーキテクト、開発者、テスター、運用担当者など、ソフトウェア開発チーム全体のインテリジェントな共同作業員として機能します。コンテキストに応じた支援を提供し、アーティファクト (ユーザーストーリー、設計モックアップ、コードスニペット、テストケースなど) を生成し、ほぼリアルタイムの提案を提供し、潜在的な問題が発生する前に予測します。この AI 拡張アプローチにより、チームメンバーの認知負荷が大幅に軽減されます。これにより、生成 AI がより日常的で反復的なタスクを処理しながら、高レベルの戦略的意思決定と複雑な問題解決に集中できます。

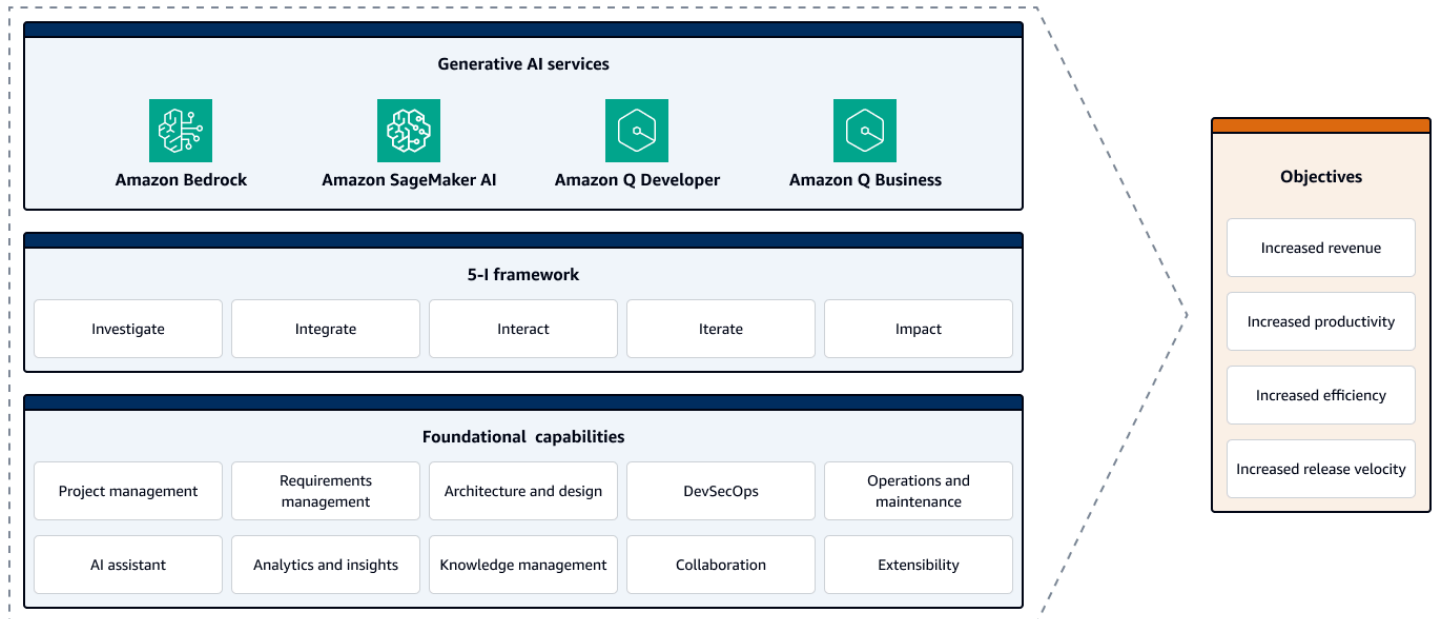
生成 AI は知識の蓄積にも役立ちます。これにより、チームメンバーは膨大なデータリポジトリから関連情報、ベストプラクティス、パターンにすばやくアクセスできます。これにより、組織全体で専門知識を効果的に民主化できます。開発ツールチェーン全体で生成 AI 機能をシームレスに統合することで、ソフトウェア開発チーム全体でより直感的で効率的で生産的な環境を作成できます。この拡張開発エクスペリエンスにより、SDLC が高速化され、全体的な品質が向上します。また、チームメンバーが新しいアイデアやアプローチをより迅速に検討できるため、エラーを減らし、イノベーションを促進します。

生成 AI を活用した開発経験を組織で採用するには、次の主要な要素を考慮してください。

- [5-I フレームワーク](#) – 5 つのディメンションで構成される 5-I フレームワーク は、最新のソフトウェア開発のプロセスをナビゲートするための包括的なアプローチを提供します。SDLC のすべてのステージに生成 AI を体系的に適用するのに役立つ構造化された方法を提供します。
- [基本的な機能](#) – 最新のソフトウェア開発のさまざまな側面で生成 AI の能力を最大限に活用するには、堅牢な基盤機能確立する必要があります。これらの機能は、AI を活用した開発エクスペリエンスのバックボーンを形成します。これらの機能は、SDLC 全体で生成 AI を統合して使用するのに役立ちます。

5-I フレームワークと基本的な機能は、ソフトウェア開発エクスペリエンスを再考するための戦略を形成します。5 つのディメンションは、生成 AI を適用するための戦略的フレームワークを提供し、基盤となる機能は、この AI 主導のアプローチをサポートする組織を準備します。 [Amazon Bedrock](#)

AWS のサービス、[Amazon SageMaker AI](#)、[Amazon Q Developer](#)、[Amazon Q Business](#) などの は、ソフトウェア開発エクスペリエンスに統合できる生成 AI の機能を提供します。



AI を活用したソフトウェア開発エクスペリエンスのための 5-I フレームワーク

5-I フレームワークは、ソフトウェア開発チームが生成 AI を開発プラクティスに効果的に統合するための構造化されたアプローチを提供します。SDLC 全体で生成 AI を使用するための堅牢な基盤を確立するのに役立ちます。また、生成 AI の可能性を最大限に活用するための適切な開発プラクティス、ワークフロー、考え方を設定するのにも役立ちます。

このセクションは、以下のトピックで構成されます。

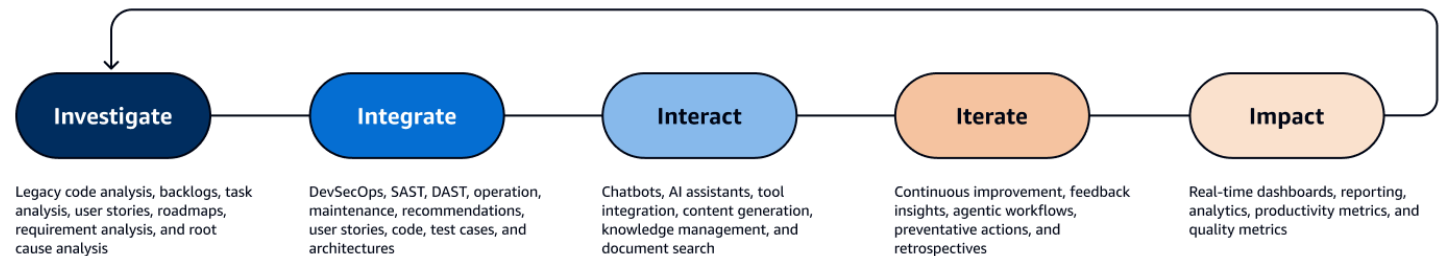
- [フレームワークの概要](#)
- [ソフトウェア開発ライフサイクルとの統合](#)

フレームワークの概要

5-I フレームワークは、調査、統合、対話、反復、影響の 5 つの主要なディメンションを中心に構築されています。各ディメンションは、生成 AI がソフトウェア開発プロセスを大幅に強化する重要な領域を表します。これらの側面にわたって生成 AI を戦略的に統合することで、フレームワークは最新のソフトウェア開発の進化するニーズに対応します。これにより、認知負荷を軽減し、創造的な可能性を高めることができます。理想的な開発エクスペリエンスはツールだけでなく、AI があらゆる

段階で人間の機能をシームレスに強化する環境を作成することにも関係していることを認識しています。

次の図は、AI を活用したソフトウェア開発の 5 つのディメンションを示しています。各ディメンションについて、効率とイノベーションを促進するために生成 AI をどこに統合できるかを示します。



フレームワークの 5 つのディメンションは次のとおりです。

- **調査** – 生成 AI を使用して、ソフトウェア開発プロセス内のすべての分析タスクを強化します。生成 AI を使用して、要件の理解、大量のデータの処理、パターンの認識、人間の能力を超えたり、生成にかなり時間がかかるインサイトの生成を行います。これらのインサイトは、情報に基づいた意思決定を行い、改善の機会をすばやく特定し、高品質のソフトウェアをより効率的に提供するのに役立ちます。生成 AI は、SDLC 全体の分析プロセスのインテリジェントなパートナーになることができます。生成 AI を活用することで、要件の収集、レガシーコードベースの検査、製品バックログの最適化など、重要な領域に詳細な分析を適用できます。たとえば、製品所有者は生成 AI を使用して、ユーザーストーリーを作成する前にユーザージャーニーや要件を分析できます。開発チームは非効率性を発見し、既存のコードベースの最適化の機会を特定できます。DevOps エンジニアは根本原因分析を適用して、パフォーマンスの問題やセキュリティの脆弱性を迅速に診断できるため、信頼性を向上させることができます。
- **統合** – 生成 AI を統合して、SDLC 全体の幅広いタスクとプロセスを自動化します。これには、コードスニペット、テストケース、アーキテクチャ設計、ユーザーストーリー、デプロイパイプラインの自動生成が含まれます。これらの通常手動タスクを自動化することで、チームはより戦略的で革新的な作業に集中できるため、市場投入までの時間を短縮し、高品質のアプリケーションを実現できます。統合ディメンションは、AI が開発プロセスの不可欠な部分となるソフトウェア開発のパラダイムシフトを表します。ソフトウェア開発チームと協力して、生産性の向上、品質の向上、イノベーションの促進を行います。これにより、市場投入までの時間が短縮されます。ソフトウェア開発チームは、各ステップで「これは自動化できますか？」と尋ねることで、プロセスとワークフローを定期的に評価する必要があります。
- **インタラクション** – 生成 AI を活用したアシスタントを使用して、さまざまなタスクやクエリで瞬時にコンテキストに応じたサポートをチームに提供します。これらのインテリジェントアシスタン

トは、膨大な情報リポジトリから派生する知識豊富な共同作業員として機能します。コーディングに関する質問への回答、設計の提案、標準運用手順の説明、複雑な問題のトラブルシューティングに役立ちます。これらの AI アシスタントを開発ワークフローに統合すると、生産性が向上し、より協力的で問題解決の環境が促進されます。

- 反復 — 生成 AI を使用して、SDLC 全体でデータ駆動型の迅速な調整を可能にします。顧客のフィードバック、使用パターン、市場傾向、チームパフォーマンスメトリクスなどのソースからのデータを継続的に分析して、情報に基づいた意思決定を迅速に行うことができます。この適応性により、ソフトウェア開発は、事前に定義された静的なプロセスから、柔軟で応答性の高いアプローチに絞り込まれます。バックログの動的な優先順位付け、柔軟なリソース割り当て、適応テスト戦略、進化するドキュメント、応答性の高いデプロイプロセスなど、さまざまな方法で現れます。たとえば、製品マネージャーは AI が生成したインサイトを使用してバックログの順序を変更し、新しい顧客要件と市場トレンドをほぼリアルタイムで統合できます。DevOps エンジニアは、パフォーマンス分析に基づいてデプロイプランとインフラストラクチャ設定を適応させ、アプリケーションの耐障害性と最適化を維持できます。開発チームは、スプリントの遡及からフィードバックを次の反復のための実用的な改善に変換し、継続的なプロセス強化の文化を推進できます。
- 影響 — 生成 AI を適用して、ソフトウェア開発プロセスの有効性とパフォーマンスを評価します。AI を活用した分析とメトリクスを使用することで、開発効率、コード品質、ユーザーエンゲージメント、アプリケーション全体のパフォーマンスに関するより深いインサイトを得ることができます。このデータ駆動型アプローチは、情報に基づいた意思決定を行い、開発ワークフローを最適化し、アプリケーションの品質とユーザーエクスペリエンスを継続的に改善するのに役立ちます。ソフトウェアチームの生産性を評価する際、生成 AI はコードのコミット頻度、問題解決時間、リリース速度、機能配信速度など、さまざまなデータポイントを分析します。また、コードレビューの品質、コラボレーションツールの有効性、チーム全体の出力に対するさまざまな開発プラクティスの影響を評価することもできます。これらのメトリクスをプロジェクトの成果に関連付けることで、AI は人間のアナリストが見逃す可能性のあるパターンや傾向を特定し、チームの生産性を高める実用的なインサイトを提供できます。さらに、生成 AI は、業界標準や履歴データに対してチームのパフォーマンスをベンチマークし、改善のためのパーソナライズされた推奨事項を提供するのに役立ちます。また、開発プロセスにおける潜在的なボトルネックやリスクを予測して、予防的な対策を講じることもできます。

ソフトウェア開発ライフサイクルとの統合

SDLC は複数のフェーズで構成され、組織ごとに異なる場合があります。一般的に、これらのフェーズには、要件と計画、設計とアーキテクチャ、実装、テスト、デプロイ、運用とメンテナンスが含まれます。

次の表は、5-I フレームワークのディメンションを SDLC フェーズにマッピングし、各ディメンションの統合レベルを示しています。

フレームワークディメンション	要件と計画	設計とアーキテクチャ	実装	テスト	デプロイメント	オペレーションとメンテナンス
調査	高	低	低	低	低	中
統合	中	中	高	中	高い	高
インタラクション	高い	高い	高	中	中	高
反復	中	低	低	低	低	中
Impact	高	中	高	低	高	高

統合のレベルは高から低までさまざまです。マッピングでは、各ディメンションの主要な重点領域が明らかになります。例えば、調査は要件と計画フェーズで高い強度を示します。統合は、実装、デプロイ、運用、メンテナンスの各フェーズで高い強度を示します。

このマッピングを使用すると、作業を効果的に優先順位付けできます。高、中、低に集中することをお勧めします。生成 AI によるソフトウェア開発エクスペリエンスを強化する、バランスの取れた影響力のあるアプローチを採用してください。

AI を活用したソフトウェア開発エクスペリエンスの基本機能

生成 AI を活用したソフトウェア開発エクスペリエンスを正常に実装するには、組織内の複数のペルソナにまたがる一連の基本的な機能確立する必要があります。これらの機能は、AI を活用したソフトウェア開発のコンテキストでリソースを効果的にデプロイし、プロセスを実装し、望ましい成果を達成する能力を表します。これらの機能を育成することで、SDLC のすべてのステージで生成 AI をシームレスに統合できる堅牢な基盤を構築できます。

AWS は、これらの機能の実装に役立つ主要なサービスを提供します。例えば、[Amazon Q Developer](#) は AI を活用したアシスタントとして機能することで、ソフトウェア開発の加速を支援します。[Amazon Q Business](#) は、差し迫った質問に対する迅速かつ適切な回答の取得、問題の解決、コンテンツの生成に役立ちます。また、ソフトウェア開発に関連するツールを統合することで、ユー

ザーに代わって動作することもできます。[Amazon Bedrock](#) は、基盤モデルとさまざまな機能にアクセスして、特定の開発ワークフローと要件をカスタマイズできます。

を通じてこれらの機能を育成することで AWS のサービス、SDLC のすべてのステージで生成 AI をシームレスに統合できる堅牢な基盤を構築できます。

以下は、重点を置くべき基本的な機能です。

- [プロジェクト管理](#)
- [要件管理](#)
- [アーキテクチャと設計](#)
- [コラボレーション](#)
- [DevSecOps](#)
- [オペレーションとメンテナンス](#)
- [AI アシスタント](#)
- [分析とインサイト](#)
- [ナレッジ管理](#)
- [拡張性](#)

各基本的な機能は、フレームワークのディメンションと SDLC のさまざまなステージと統合されます。この統合により、ソフトウェア開発プロセス全体で AI 機能を効果的に使用できます。これにより、すべてのステップで効率、品質、イノベーションが向上します。これらの基本的な機能、フレームワーク、および SDLC ステージ間のシナジーにより、AI を活用したソフトウェア開発のための包括的なエコシステムが作成されます。これにより、生成 AI の可能性を最大限に活用し、継続的な改善を推進し、開発サイクルを加速し、高品質のソフトウェア製品を提供できます。

次の表は、基本的な機能とサブ機能がフレームワークディメンションと SDLC フェーズにどのようにマッピングされるかを示しています。

機能: サブ機能	調査	統合	操作	反復	Impact
プロジェクト管理: 問題管理	要件と計画	なし	なし	なし	なし

機能: サブ機能	調査	統合	操作	反復	Impact
プロジェクト管理: スプリントとタスクの管理	要件と計画	要件と計画	なし	なし	なし
プロジェクト管理: 製品バックログ管理	要件と計画	なし	なし	要件と計画	なし
プロジェクト管理: ユーザーストーリーマッピング	要件と計画	なし	なし	なし	なし
プロジェクト管理: レポートと分析	要件と計画	なし	なし	なし	要件と計画
プロジェクト管理: 製品ロードマップ管理	要件と計画	なし	要件と計画	なし	なし
プロジェクト管理: フィードバックループ	なし	なし	なし	要件と計画	なし
プロジェクト管理: 遡及的	なし	なし	なし	要件と計画	なし
要件管理	要件と計画	要件と計画	なし	なし	なし

機能: サブ機能	調査	統合	操作	反復	Impact
アーキテクチャと設計: ソリューション設計	設計とアーキテクチャ	設計とアーキテクチャ	なし	なし	なし
コラボレーション: ドキュメント管理	すべての SDLC フェーズ	なし	すべての SDLC フェーズ	なし	なし
コラボレーション: ナレッジ共有	すべての SDLC フェーズ	なし	すべての SDLC フェーズ	なし	なし
コラボレーション: プロジェクトアセット管理	なし	すべての SDLC フェーズ	すべての SDLC フェーズ	なし	なし
DevSecOps: CI/CD	テスト、デプロイ	実装、テスト、デプロイ	デプロイメント	なし	なし
DevSecOps : DevOps セキュリティ	実装	実装、テスト、運用、メンテナンス	なし	実装、テスト、運用、メンテナンス	なし
DevSecOps : アプリケーションパフォーマンスのモニタリング	なし	オペレーションとメンテナンス	なし	なし	なし

機能: サブ機能	調査	統合	操作	反復	Impact
DevSecOps: ログの集約と分析	オペレーションとメンテナンス	オペレーションとメンテナンス	なし	なし	なし
DevSecOps: AIOps	オペレーションとメンテナンス	なし	なし	オペレーションとメンテナンス	なし
DevSecOps: 継続的な改善	なし	なし	なし	オペレーションとメンテナンス	なし
DevSecOps: ダッシュボードのモニタリング	なし	オペレーションとメンテナンス	なし	なし	なし
DevSecOps: パフォーマンスインサイト	オペレーションとメンテナンス	なし	なし	オペレーションとメンテナンス	なし
運用とメンテナンス: インシデント管理	なし	なし	なし	オペレーションとメンテナンス	なし
オペレーションとメンテナンス: コードのアップグレード	なし	オペレーションとメンテナンス	なし	なし	なし
オペレーションとメンテナンス: コードの最適化	オペレーションとメンテナンス	オペレーションとメンテナンス	なし	なし	なし

機能: サブ機能	調査	統合	操作	反復	Impact
運用とメンテナンス: 技術的負債管理	なし	オペレーションとメンテナンス	オペレーションとメンテナンス	なし	なし
運用とメンテナンス: 変更管理	なし	実装、デプロイ	なし	なし	なし
運用とメンテナンス: リバースエンジニアリング	オペレーションとメンテナンス	なし	なし	なし	なし
運用とメンテナンス: コードのモダライゼーション	なし	実装	なし	なし	なし
運用とメンテナンス: パフォーマンスの最適化	なし	オペレーションとメンテナンス	なし	オペレーションとメンテナンス	なし
分析とインサイト	なし	要件と計画	なし	なし	すべての SDLC フェーズ
AI アシスタント	なし	なし	すべての SDLC フェーズ	なし	なし
ナレッジ管理	なし	なし	すべての SDLC フェーズ	なし	なし

機能: サブ機能	調査	統合	操作	反復	Impact
拡張性	なし	デプロイメント	なし	なし	なし

プロジェクト管理のための生成 AI ユースケース

効果的なプロジェクト管理は、ソフトウェア開発の成功の中心です。生成 AI のコンテキストでは、プロジェクト管理は新しいディメンションを取ります。より予測的、適応的、データ駆動型になる可能性があります。AI を活用したプロジェクト管理ツールは、過去のプロジェクトデータを分析して、より正確な時間とリソースの見積もりを生成します。ビジネス目標とチームのキャパシティに基づいてタスクに自動的に優先順位を付けることができ、潜在的な障害が発生する前に予測することもできます。例えば、プロジェクトマネージャーは生成 AI を使用して、プロジェクトの要件と同様のプロジェクトの履歴データに基づいて予備プロジェクト計画を作成できます。AI は、スキル、ワークロード、プロジェクトのニーズを考慮した最適なチーム構成を提案できます。プロジェクト全体で AI 駆動型ダッシュボードは、レポートを自動的に生成し、注意が必要な領域を強調することで、プロジェクトのステータスに関するほぼリアルタイムのインサイトを提供します。

プロジェクト管理に対するこの AI 拡張アプローチは、効率を高めることができます。これは、プロジェクトマネージャーが日常的な管理タスクに没頭するのではなく、戦略的意思決定とチームリーダーシップに集中するのに役立ちます。

次の表は、生成 AI で強化できるプロジェクト管理のユースケースと、それらのユースケースを担当するペルソナを示しています。

サブ機能: ユースケース	ペルソナ
問題管理: 問題を作成して割り当てる	プロジェクトマネージャー
問題管理: テスト中に問題を検出し、ログに記録する	テストエンジニア
問題管理: 重要度に基づいて問題を優先順位付けし、デベロッパーに割り当てる	プロジェクトマネージャー
問題管理: 重複する問題を特定してマージする	プロジェクトマネージャー

サブ機能: ユースケース	ペルソナ
問題管理: プロジェクトの主要な問題、メトリクス、全体的な状態に関するレポートを追跡して生成します。	プロジェクトマネージャー
スプリントとタスクの管理: タスクの労力を見積もり、チームのキャパシティに基づいてストーリーポイントを割り当てる	スクラムマスター
スプリントとタスクの管理: チームメンバー間でタスクを分散し、スプリント全体にワークロードを均等に分散します。	スクラムマスター
スプリントとタスクの管理: チームの取り組みをスプリントの目標と整合させるスプリント計画セッションを促進する	スクラムマスター
製品バックログ管理: ビジネス価値、緊急性、ユーザーフィードバックに基づいてバックログ項目を並べ替える	プロダクトオーナー
製品バックログ管理: 新しい顧客のフィードバックと市場インサイトを製品バックログに統合して、ほぼリアルタイムの優先順位付けを行います。	プロダクトオーナー
製品バックログ管理: バックログ項目間の依存関係を特定および管理して開発を効率化する	製品マネージャー
ユーザーストーリーマッピング: ユーザージャーニーのマップを作成して、必要なすべての機能とそれに対応するユーザーストーリーを特定します。	プロダクトオーナー
ユーザーストーリーマッピング: ユーザーフローのギャップや欠落しているステップを特定する	ビジネスアナリスト

サブ機能: ユースケース	ペルソナ
ユーザーストーリーマッピング: ビジネス価値への影響に基づいてユーザーストーリーを優先する	製品マネージャー
レポートと分析: スプリント速度や問題解決率などの主要なプロジェクトメトリクスを視覚化するほぼリアルタイムのダッシュボードを生成する	プロジェクトマネージャー
レポートと分析: 履歴データを分析し、潜在的な遅延やボトルネックなど、将来のプロジェクト成果を予測する	プロジェクトマネージャー
レポートと分析: チームのパフォーマンスやプロジェクトのステータスレポートなど、さまざまなステークホルダーに合わせたカスタムレポートを作成します。	プロジェクトマネージャー
製品ロードマップ管理: 主要なマイルストーンとリリース日の概要を示す製品ロードマップを作成および維持する	プロジェクトマネージャー
製品ロードマップ管理: プロジェクトの優先順位やタイムラインの変更に基づいてロードマップを更新する	製品マネージャー
製品ロードマップ管理: ステークホルダーとロードマップを共有し、製品の方向性を可視化する	製品マネージャー
フィードバックループ: 各スプリント後にチームからフィードバックを収集し、改善すべき分野を特定する	スクラムマスター
遡及的: フィードバックを実行可能な項目に変換して次のスプリントを行い、継続的な改善を促進します。	スクラムマスター

アーキテクチャと設計のための生成 AI ユースケース

プロジェクト管理の強固な基盤と明確に定義された要件により、次の重要な機能はアーキテクチャと設計です。ここでは、生成 AI によって、堅牢でスケーラブル、効率的なソフトウェアアーキテクチャを作成するための新しい可能性が開かれています。AI を活用した設計ツールは、要件と制約を分析して、最適なアーキテクチャパターンと設計アプローチを提案できます。複数の設計代替案を生成し、それぞれがパフォーマンス、スケーラビリティ、保守性など、さまざまな優先順位に合わせて最適化されます。たとえば、ソリューションアーキテクトは AI アシスタントを使用して、プロジェクト要件に基づいていくつかの高レベルのアーキテクチャ設計をすばやく生成できます。この AI 拡張アプローチは、設計プロセスを加速し、アーキテクトがより多くの情報に基づいた意思決定を行うのに役立ちます。これにより、より堅牢で将来性のあるソフトウェア設計が可能になります。

次の表は、生成 AI で強化できるアーキテクチャと設計のユースケースと、それらのユースケースを担当するペルソナを示しています。

ユースケース	ペルソナ
アーキテクチャドキュメントを作成する	ソリューションアーキテクト
詳細な設計ドキュメントを作成する	テクニカルリード
既存のアーキテクチャと設計標準を理解する	ソリューションアーキテクト
ユーザーインターフェイスの詳細なモックアップとプロトタイプを開発する	UX/UI デザイナー

コラボレーションの生成 AI ユースケース

ソフトウェア開発は本質的に共同作業です。生成 AI を使用して、ソフトウェア開発チームのコラボレーションを強化できます。AI を活用したコラボレーションツールは、シンプルなメッセージングやファイル共有にとどまりません。長いディスカッションスレッドを要約し、主要な決定事項を強調し、チームメンバーのスケジュールと生産性パターンに基づいて会議に最適な時間を提案することで、より効果的なコミュニケーションを促進します。AI は、潜在的な問題を自動的に特定し、改善を提案し、レビュワーに複雑な変更を説明することで、コードレビューを支援できます。ブレインストーミングセッション中、AI は進行役として機能し、アイデアを生成し、考えを整理し、さらには議論を仲介して、すべての音声が確実に聞こえるようにすることができます。分散型チームにとって、AI は文化や言語の障壁を埋めるのに役立ちます。チャットやビデオ通話でほぼリアルタイムの言語翻訳を提供し、誤解を防ぐために文化的コンテキストを提供できます。AI との人間によるコラ

ボレーションを強化することで、チームはより効率的かつ効果的に作業できるため、イノベーションが促進され、プロジェクト全体の成果が向上します。

次の表は、生成 AI を使用してコラボレーションのユースケースを強化する方法を示しています。

サブ機能: ユースケース	ペルソナ
ドキュメント管理: 一元化されたドキュメントリポジトリを作成して維持する	テクニカルライター
ドキュメント管理: 複数のチームメンバーがリアルタイムでドキュメントを共同作業できるようにする	開発チーム
ナレッジ共有: 開発者が協力して質問し、知識を共有し、問題をトラブルシューティングするためのプラットフォームとしてディスカッションフォーラムを使用する	開発チーム
ナレッジ共有: ディスカッションフォーラムを使用して、プロジェクトのディスカッション中に行われた決定を文書化して追跡し、重要な決定の根拠がキャプチャされ、将来の参照のためにアクセス可能であることを確認します。	製品マネージャー
プロジェクトアセット管理: プロジェクト関連リソースの共有を容易にする	開発チーム
プロジェクトアセット管理: 共有コンテンツのバージョン管理を実装して、チームメンバーが変更を追跡し、以前のバージョンに戻して、コンテンツ更新でコラボレーションできるようにします。	開発チーム

DevSecOps の生成 AI ユースケース

AI を活用した DevSecOps ツールは、ソフトウェア配信パイプラインの多くの側面を自動化します。たとえば、インテリジェントなコードレビューの実行、潜在的なバグの検出、セキュリティの脆弱

弱性の検出、デベロッパーがコードを記述する際のパフォーマンスの問題のほぼリアルタイムの特定を行うことができます。AI は包括的なテストスイートを生成して実行し、コードベースの進化に合わせて自動的に更新します。DevSecOps に対するこの AI 拡張アプローチは、配信パイプラインを加速し、配信されるソフトウェアのセキュリティと信頼性を大幅に強化します。

次の表は、生成 AI で強化できる DevSecOps ユースケースと、それらのユースケースを担当するペルソナを示しています。

サブ機能: ユースケース	ペルソナ
DevOps と継続的デリバリー: デプロイパイプライン全体を自動化	DevOps エンジニア
DevOps と継続的デリバリー: コードの品質と潜在的な問題に関するほぼリアルタイムのフィードバックを受け取る	ソフトウェア開発者
DevOps と継続的デリバリー: ほぼリアルタイムのセキュリティ問題と修復の推奨事項を受け取る	ソフトウェア開発者
DevOps と継続的デリバリー: ほぼリアルタイムのコードとベストプラクティスの提案を受け取る	ソフトウェア開発者
DevOps と継続的デリバリー: 反復タスクを自動化し、コマンドをスクリプトに統合する	DevOps エンジニア
DevOps と継続的デリバリー: コードを作成し、各コードコミット後にアーティファクトを自動的に生成する	ソフトウェア開発者
DevOps と継続的デリバリー: 組織の標準とフレームワークに従ってコードを構築する	ソフトウェア開発者
DevOps と継続的デリバリー: すべてのコミットでユニットテストを自動的に実行して、開発プロセスの早い段階でエラーをキャッチする	ソフトウェア開発者

サブ機能: ユースケース	ペルソナ
DevOps と継続的デリバリー: ユニットテストのカバレッジを分析し、すべての重要なコードパスがテストされていることを確認します。	ソフトウェア開発者
DevOps と継続的デリバリー: ブランチを管理し、変更をマージする	ソフトウェア開発者
DevOps と継続的デリバリー: コードとアーティファクトのバージョンングを管理する	ソフトウェア開発者
DevOps と継続的デリバリー: ビルドアーティファクトと依存関係の保存と管理	DevOps エンジニア
DevOps と継続的デリバリー: ビルドプロセス中に依存関係を解決して取得する	ソフトウェア開発者
DevOps と継続的デリバリー: 統合テストを生成して実行し、コンポーネントが期待どおりに連携することを確認します。	テストエンジニア
DevOps と継続的デリバリー: 統合テスト中にモックサービスを使用して外部システムとのインタラクションをシミュレートする	テストエンジニア
DevOps と継続的デリバリー: さまざまな負荷でアプリケーションのパフォーマンスをベンチマークする	パフォーマンスエンジニア
DevOps と継続的デリバリー: トラフィックの多いシナリオをシミュレートしてアプリケーションのスケーラビリティと応答時間をテストする	パフォーマンスエンジニア
DevOps と継続的デリバリー: サーバーのクラッシュやネットワーク停止などの障害から回復するシステムの機能をテストする	サイト信頼性エンジニア

サブ機能: ユースケース	ペルソナ
DevOps と継続的デリバリー: カオスエンジニアリングを実行する	サイト信頼性エンジニア
DevOps と継続的デリバリー: テストを実行して、アプリケーションがビジネス要件を満たしていることを確認します。	QA エンジニア
DevOps と継続的デリバリー: ユーザー受け入れテストを実施する	プロダクトオーナー
DevOps と継続的デリバリー: 脆弱性とライセンスコンプライアンスの問題の依存関係をスキャンします	セキュリティエンジニア
DevOps と継続的デリバリー: オープンソースの依存関係を監視および管理して、それらが最新で安全であることを確認します。	セキュリティエンジニア
DevOps と継続的デリバリー: ソフトウェア部品表 (SBOM) を生成して維持し、すべてのコンポーネントと依存関係を追跡する	セキュリティエンジニア
DevOps と継続的デリバリー: SBOM を使用して規制コンプライアンスの監査を実施する	コンプライアンス責任者
DevOps と継続的デリバリー: リリースノートを作成する	リリースマネージャー
DevOps と継続的デリバリー: リリースの計画と調整	リリースマネージャー
DevOps と継続的デリバリー: ロールバックとリリース管理の標準運用手順を実装する	リリースマネージャー
DevOps と継続的デリバリー: 機能フラグを使用して、新しいコードをデプロイせずに本番環境の機能を有効化または無効化する	製品マネージャー

サブ機能: ユースケース	ペルソナ
DevOps と継続的デリバリー: 機能フラグを使用して A/B テストを実行し、さまざまな機能がユーザー動作に与える影響を測定します。	製品マネージャー
DevOps と継続的デリバリー: パイプラインの障害の分析とモニタリング	DevOps エンジニア
DevOps と継続的デリバリー: インフラストラクチャリソースの作成と管理	DevOps エンジニア
DevOps とセキュリティ: ハードコードされたシークレットのコードリポジトリをスキャンします。	DevOps エンジニア
DevOps とセキュリティ: ほぼリアルタイムの検出を実装して、シークレットがリポジトリにコミットされた場合にすぐに開発者に警告する	DevOps エンジニア
DevOps とセキュリティ: 継続的なコード品質モニタリングを実施する	ソフトウェア開発者
DevOps とセキュリティ: コード内の潜在的なセキュリティ脆弱性のインジケータを検出してフラグ付けする	ソフトウェア開発者
DevOps とセキュリティ: Open Worldwide Application Security Project (OWASP) の上位 10 のセキュリティリスクの自動テストを実装し、アプリケーションが業界標準のセキュリティプラクティスに準拠していることを確認します。	セキュリティエンジニア
DevOps とセキュリティ: 開発プロセスにチェックを統合することで、OWASP リスクについてデベロッパーを定期的に更新して教育する	セキュリティエンジニア

サブ機能: ユースケース	ペルソナ
DevOps とセキュリティ: サードパーティーのライブラリと依存関係をスキャンして既知のセキュリティの脆弱性を検出する	DevOps エンジニア
DevOps とセキュリティ: アプリケーションコードとインフラストラクチャをスキャンして脆弱性を検出する	DevOps エンジニア
DevOps とセキュリティ: デプロイ前にコードの脆弱性を分析する	セキュリティエンジニア
DevOps とセキュリティ: 重大な脆弱性を持つコードがマージされないようにしてセキュリティポリシーを適用する	セキュリティエンジニア
DevOps とセキュリティ: ロールベースのアクセスコントロール (RBAC) を実装して、機密性の高いシステムやデータへのアクセスを制限し、承認された担当者のみが重要なリソースにアクセスできるようにします。	セキュリティエンジニア
DevOps とセキュリティ: チーム構造の変化に適応することで、役割と責任に基づいてアクセスコントロールを調整する	DevOps エンジニア
DevOps とセキュリティ: 本番環境への攻撃をシミュレートして、実行中のアプリケーションをほぼリアルタイムでテストします。	セキュリティエンジニア
DevOps とセキュリティ: デプロイされたアプリケーションにセキュリティの脆弱性を継続的にモニタリングする	DevOps エンジニア
DevOps とセキュリティ: すべての環境で定期的な脆弱性スキャンをスケジュールして、セキュリティの弱点を特定して対処する	セキュリティエンジニア

サブ機能: ユースケース	ペルソナ
DevOps とセキュリティ: 脆弱性スキャンの結果に基づいてパッチと更新を適用し、安全なシステムを維持するのに役立ちます。	DevOps エンジニア
アプリケーションパフォーマンスのモニタリング: アプリケーションパフォーマンスをほぼリアルタイムで継続的にモニタリングし、ユーザーに影響を与える前にパフォーマンスの問題を検出して診断します。	サイト信頼性エンジニア
アプリケーションパフォーマンスのモニタリング: 応答時間の急増やエラー率の増加などのパフォーマンスの異常を検出し、アラートを開始します。	DevOps エンジニア
アプリケーションパフォーマンスのモニタリング: 分散システムを介して伝達されるリクエストをトレースして、パフォーマンスのボトルネックとレイテンシーの問題を特定します。	DevOps エンジニア
アプリケーションパフォーマンスのモニタリング: 分散トレースを使用して、障害やパフォーマンスの低下の原因となっているサービスやコンポーネントを正確に特定する	DevOps エンジニア
ログの集約と分析: 複数のソースからのログを一元化されたシステムに集約し、傾向と問題を特定するための検索と分析を容易にします。	サイト信頼性エンジニア
ログの集約と分析: 自動ログ解析を実装して関連情報を抽出し、問題を示す可能性のあるパターンや異常を検出します。	DevOps エンジニア
ログの集約と分析: 主要なパフォーマンスメトリクスを収集して視覚化する	サイト信頼性エンジニア

サブ機能: ユースケース	ペルソナ
ログの集約と分析: 事前定義されたサービスレベルアグリーメント (SLAs)	製品マネージャー
AI オペレーション: インシデントを検出し、根本原因を分析し、人間の介入なしに是正措置を開始する	DevOps エンジニア
AI オペレーション: 停止を回避するために、将来のリソース需要を予測し、キャパシティプランニングを最適化する	サイト信頼性エンジニア
継続的な改善: アプリケーションとの実際のユーザーインタラクションをモニタリングして、パフォーマンスに関するインサイトを収集し、改善すべき分野を特定する	UX デザイナー
継続的な改善: さまざまな地理的リージョンでアプリケーションのパフォーマンスを追跡し、一貫したユーザーエクスペリエンスをグローバルに確保する	製品マネージャー
ダッシュボードモニタリング: カスタマイズ可能なダッシュボードを作成して重要なメトリクス、ログ、トレースをほぼリアルタイムで視覚化し、システムの状態を包括的に把握できます。	サイト信頼性エンジニア
ダッシュボードモニタリング: さまざまなチーム (開発チーム、運用チーム、製品チームなど) のダッシュボードを作成し、重点分野に基づいて関連するインサイトを提供します。	DevOps エンジニア
パフォーマンスインサイト: アプリケーションパフォーマンスの詳細な分析を実施して非効率性を特定し、コードまたはインフラストラクチャを最適化する	ソフトウェア開発者

サブ機能: ユースケース	ペルソナ
パフォーマンスインサイト: パフォーマンスインサイトを使用してアプリケーションパフォーマンスを繰り返し改善し、時間の経過とともにユーザーエクスペリエンスを最適化する	製品マネージャー

運用とメンテナンスのための生成 AI ユースケース

ソフトウェアがデプロイされると、フォーカスはオペレーションとメンテナンスに移行します。生成 AI は、よりプロアクティブで効率的なシステム管理を提供することで、従来のアプローチを強化できます。AI を活用した運用ツールは、システムパフォーマンスを継続的にモニタリングし、潜在的な問題をユーザーに影響を与える前に予測します。問題が発生したときに自動根本原因分析を実行するため、解決までの平均時間が大幅に短縮されます。AI は、システムパフォーマンスをほぼリアルタイムで最適化します。負荷パターンとユーザー動作の変化に基づいて設定を自動的に調整します。たとえば、運用チームは AI アシスタントを使用して予測メンテナンススケジュールを生成し、失敗する可能性のあるコンポーネントを自動的に特定し、予防アクションを提案する場合があります。AI は、使用状況の傾向を分析し、将来のリソースニーズを高精度に予測することで、キャパシティプランニングにも役立ちます。

次の表は、生成 AI で強化できるオペレーションとメンテナンスのユースケースと、それらのユースケースを担当するペルソナを示しています。

サブ機能: ユースケース	ペルソナ
インシデント管理: モニタリングツールをチャットプラットフォームと統合してインシデントをほぼリアルタイムで管理し、チームがチャット環境内で直接問題を検出、議論、解決できるようにします。	サイト信頼性エンジニア
インシデント管理: チームがチャットインターフェイスから直接デプロイを開始し、スクリプトを実行し、コマンドを実行できるようにします。これにより、オペレーションが効率化されます。	DevOps エンジニア

サブ機能: ユースケース	ペルソナ
コードのアップグレード: コードの依存関係とライブラリをアップグレードして手動作業を減らし、コードベースが最新バージョンで最新の状態を維持できるようにします。	ソフトウェア開発者
コードの最適化: 最適化の機会のコードを確認する	ソフトウェア開発者
コードの最適化: コードのボトルネックを特定し、リファクタリングまたは最適化してパフォーマンスを向上させる	ソフトウェア開発者
技術的負債管理: 開発プロセスの一環として技術的負債を記録する	製品マネージャー
技術的負債管理: 影響、リスク、コストに基づいて技術的負債を優先順位付けして対処し、通常のスプリント計画プロセスに統合します。	ソフトウェア開発者
技術的負債管理: 既存のアプリケーションコードの技術的負債の削減	ソフトウェア開発者
変更管理: すべてのコード変更がデプロイ前に必要なステークホルダーによってレビュー、テスト、承認されていることを確認する変更承認プロセスを実装します。	変更マネージャー
変更管理: 提案された変更の影響分析を実行する	DevOps エンジニア
リバースエンジニアリング: レガシーコードの構造と動作を分析して理解する	ソリューションアーキテクト
リバースエンジニアリング: 既存のコードの説明とドキュメントの生成	ソフトウェア開発者

サブ機能: ユースケース	ペルソナ
コードモダナイゼーション: あるプログラミング言語から別のプログラミング言語にコードを翻訳する	ソフトウェア開発者
コードモダナイゼーション: レガシーコードを最新のプログラミング言語にモダナイズする	ソフトウェア開発者
パフォーマンスの最適化: リソースの割り当て、負荷分散、アプリケーションの再設定を最適化して、システムパフォーマンスを継続的にモニタリングおよび調整する	サイト信頼性エンジニア
パフォーマンスの最適化: パフォーマンスの低下を引き起こしているコードを特定してリファクタリングし、速度とシステムの応答性を向上させます。	ソフトウェア開発者

ソフトウェア開発における生成 AI アシスタントのユースケース

AI アシスタント機能は、生成 AI を活用した開発エクスペリエンスの中心です。このインテリジェントなコンテキスト対応システムは、SDLC 全体のすべてのチームメンバーの仮想共同作業員として機能します。複雑なコードを扱う開発者を想像してみてください。AI アシスタントに支援を求めるだけで、関連するコードスニペットを提供したり、複雑なアルゴリズムを説明したり、現在のコンテキストやベストプラクティスに基づいて最適化を提案したりできます。AI アシスタントは、ITOps マネージャーが内部ドキュメントに基づく標準運用手順を理解するのに役立ちます。AI アシスタントは、コンテキストに応じた即時サポートを提供することで、チームメンバーの認知負荷を大幅に軽減します。これにより、より高いレベルの問題解決と創造的なタスクに集中できます。この機能は、ソフトウェア開発のすべての段階で生産性と品質を向上させる力の乗数として機能します。

次の表は、AI アシスタントとメリットのあるペルソナで強化できるユースケースを示しています。

ユースケース	ペルソナ
要件、アーキテクチャ、標準運用手順などの質問に答えて、開発チームにすぐに支援を提供する	ソフトウェア開発チーム
自然言語クエリを使用して、広範なドキュメントから抜粋を検索または取得するか、概要を生成する	ソフトウェア開発チーム
要件ドキュメント、アーキテクチャ設計ドキュメント、内部プロセスなど、長い技術ドキュメントを要約する	ソフトウェア開発チーム
チームが一般的なタスクに使用できるプロンプトのライブラリを維持する	ソフトウェア開発チーム
生成 AI を既存のツールやシステムにシームレスに統合する	ソフトウェア開発チーム
さまざまなプラットフォーム、ツール、内部システムにわたるタスクを自動化する	ソフトウェア開発チーム
ベストプラクティス、プロジェクト固有の情報、チームの知識など、すべてのチームメンバーがアクセスできる一元的な知識リポジトリを作成します。	ソフトウェア開発チーム
タスクのコンテキストに基づいてリポジトリから関連する知識を取得する	ソフトウェア開発チーム
自動コードレビュー、根本原因分析の実行、改善の提案、潜在的なバグの検出、トラブルシューティングの実行	ソフトウェア開発者、DevOps エンジニア、サイト信頼性エンジニア
パフォーマンスデータを分析して、パフォーマンスの最適化に関する意思決定に役立つ傾向とパターンを特定する	サイト信頼性エンジニア

ユースケース	ペルソナ
効率の向上、複雑さの軽減、セキュリティの強化に関する推奨事項を提供する	ソフトウェア開発者
スケーリングの推奨事項やコスト削減戦略など、クラウドリソースの使用の最適化を提案する	ソフトウェア開発者、DevOps エンジニア、サイト信頼性エンジニア、ソリューションアーキテクト
コード、ユーザーガイド、製品機能リリースに基づくドキュメントなどの新しいコンテンツを生成する	ソフトウェア開発チーム

分析とインサイトの生成 AI ユースケース

分析とインサイトの機能は、膨大な量のデータを実用的なインサイトに変換し、意思決定と継続的な改善を促進するのに役立ちます。生成 AI を使用することで、この機能はコードリポジトリ、プロジェクト管理ツール、チームコラボレーションプラットフォームなど、さまざまなソースからのデータを処理し、開発プロセスとチームの生産性を包括的に把握できます。生成 AI は、予測分析と規範分析を提供するために、従来のメトリクスを超えています。潜在的な問題を予測し、ターゲットを絞った改善を提案できます。たとえば、コードコミット、バグ解決率、機能配信速度のパターンを分析して、パフォーマンスの高いチームを特定し、ボトルネックを特定し、プロセスの最適化を提案できます。さらに、チームのダイナミクスと個々のパフォーマンスに関するインサイトを提供できます。これらのインサイトは、リーダーがワークロードの分散、トレーニングニーズ、チーム構成に関するデータ主導型の意思決定を行うのに役立ちます。インタラクティブダッシュボードを通じてこれらのインサイトを提示することで、あらゆるレベルのステークホルダーが情報に基づいた意思決定を行い、プロセスを最適化し、チームの生産性を継続的に向上させ、高品質のソフトウェアを迅速に提供できるようになります。

次の表は、生成 AI で強化できる分析ユースケースと、それらのユースケースを担当するペルソナを示しています。

ユースケース	ペルソナ
個人とチームの生産性をモニタリングする	開発マネージャー

ユースケース	ペルソナ
生産性の傾向を分析して潜在的なバーンアウトを検出し、チームのウェルビーイングと生産性を維持するために積極的な対策を講じることができます。	開発マネージャー
コード変更が本番環境にデプロイされる頻度を追跡し、開発プロセスの速度と俊敏性を測定します。	製品マネージャー
デプロイ頻度データを分析して、プロセスの非効率性やリソースの制約を示す可能性のあるデプロイアクティビティが少ない期間を特定する	製品マネージャー
開発とデプロイプロセスを合理化する機会を特定するために、コードがデプロイにコミットされるまでの時間を測定します。	開発マネージャー
リリースプロセスの信頼性を評価するために、即時修復が必要な障害につながるデプロイの割合を追跡する	サイト信頼性エンジニア
変更失敗率メトリクスを使用して、頻繁に問題を引き起こすコード領域を特定し、ターゲットを絞ったリファクタリングとテスト作業をガイドする	ソフトウェア開発者
停止またはインシデント後のサービスの復元にかかる時間をモニタリングして、ダウンタイムを短縮し、システム全体の耐障害性を向上させることができます。	サイト信頼性エンジニア
復旧時間の傾向を分析してインシデント対応プロセスを強化し、システム障害からの復旧を迅速化	DevOps エンジニア

ユースケース	ペルソナ
開発と運用の健全性を包括的に把握するために、デプロイ頻度、リードタイム、変更失敗率などの主要なメトリクスを集約するカスタマイズされたダッシュボードを作成する	製品マネージャー
開発、運用、ビジネスなど、特定の責任分野に焦点を当てたインサイトを提供するために、さまざまなチームのニーズに合わせたダッシュボードを作成する	製品マネージャー
収益への影響、顧客満足度、市場シェアなどのビジネス主要業績評価指標 (KPIs) を追跡し、開発の取り組みをより広範なビジネス目標に合わせる	製品マネージャー
ビジネス KPIs で、その成功を評価し、将来の製品開発をガイドする	ビジネスアナリスト
コードベースが維持可能で安全であることを確認するために、コードの複雑さ、テストカバレッジ、バグ密度などのコード品質メトリクスをモニタリングする	ソフトウェア開発者
長期的な持続可能性を推進し、技術的負債を減らすためにリファクタリングが必要なコードベースの領域を特定する	ソリューションアーキテクト

ナレッジ管理のための生成 AI ユースケース

ソフトウェア開発組織では、知識は重要なアセットです。生成 AI を活用したナレッジ管理機能により、このアセットのキャプチャ、整理、使用方法が強化されます。従来のナレッジ管理システムでは、情報が多すぎたり、古いコンテンツが含まれていたり、関連情報をすばやく見つけるために検索するのが難しいことがよくあります。

生成 AI は、これらの課題に正面から対処します。コードの変更、会話、プロジェクトアーティファクトに基づいてドキュメントを自動的に生成および更新します。これにより、チームメンバーの手

動作業を必要とせずに、ナレッジベースを最新の状態に保つことができます。さらに重要なのは、AI はこの知識を直感的な方法でアクセスできるようにすることです。チームメンバーは自然言語で質問でき、AI は関連する回答を提供できます。AI は、公式ドキュメント、コードコメント、ディスカッションスレッド、外部リソースなど、さまざまなソースから取得できます。たとえば、特定のコンポーネントを理解しようとする新しいチームメンバーが AI に「認証モジュールの仕組み」と尋ねることがあります。AI は、簡潔な説明と、関連するコードセクション、アーキテクチャ図、最近の変更へのリンクを提供します。チームメンバーの役割と専門知識のレベルに基づいてこの情報を調整することもできます。

この機能により、オンボーディングが高速化され、反復的な質問が軽減され、組織全体の知識共有が促進されます。組織の知識を保持するため、チームは時間の経過とともに複雑なシステムを簡単に維持および進化できます。

次の表は、生成 AI で強化できるナレッジ管理のユースケースと、それらのユースケースを担当するペルソナを示しています。

ユースケース	ペルソナ
プロジェクト関連のすべての知識に簡単にアクセスできる統合プラットフォームを作成する	ソフトウェア開発チーム
さまざまな開発アクティビティから知識をキャプチャする	ソフトウェア開発チーム
リポジトリ内の関連する知識をすばやく見つけるための高度な検索機能を提供する	ソフトウェア開発チーム
チームの学習モジュールとパスをパーソナライズする	ソフトウェア開発チーム

拡張のための生成 AI ユースケース

拡張性により、組織は AI システムを特定のニーズに合わせて調整しながら、既存のツールやワークフローとシームレスに統合できます。この機能は、一般的な開発およびプロジェクト管理ツールへの AI 機能の統合を容易にする堅牢な APIs、SDKs、カスタマイズ可能なインターフェイスを提供します。たとえば、組織は AI を活用した機能を使用して Jira を強化し、チケットの優先順位付け、労力の見積もり、スプリント計画を自動化できます。Jenkins パイプラインを AI で拡張して、インテリジェントなビルドの最適化と予測テストの選択を行うことができます。

さらに、拡張性により、統合開発環境 (IDEs)、バージョン管理システム、コードレビュープラットフォームとの深い統合が可能になります。AI は、コードの作成、コードレビューの自動化、コンテキストドキュメントの生成に役立ちます。

この機能は、組織固有のデータに関する AI モデルのトレーニングと微調整もサポートしています。これにより、AI は会社固有のコーディングパターン、アーキテクチャの好み、ドメインの知識を理解できます。結果は、すべての統合ツールでより関連性が高く、コンテキストに応じた支援を提供します。このレベルの柔軟性と統合を提供することで、拡張性により、AI を活用した開発エクスペリエンスが組織とともに進化します。既存のツールチェーンやワークフローをシームレスに強化しながら、変化するテクノロジーやビジネスニーズに適応できます。

次の表は、生成 AI で強化できる拡張性のユースケースと、それらのユースケースを担当するペルソナを示しています。

ユースケース	ペルソナ
サードパーティーのツールを開発環境に統合する	DevOps エンジニア
チーム固有の開発プロセスに合わせてカスタマイズされたカスタムオートメーションワークフローを作成する	DevOps エンジニア
さまざまな APIs とサービスに接続する	DevOps エンジニア
クロスプラットフォームツール用のコネクタを作成する	DevOps エンジニア

ソフトウェア開発で生成 AI を使用するためのベストプラクティス

このセクションでは、生成 AI をソフトウェア開発ライフサイクル (SDLC) に統合するためのベストプラクティスについて説明します。これらのガイドラインは、シームレスなツールチェーンと DevSecOps パイプラインの実装からコラボレーションの促進や反復タスクの自動化まで、AI の能力を活用して開発プロセスとエクスペリエンスを向上させるのに役立ちます。これらのベストプラクティスに従うことで、ソフトウェア開発チームは作業における新しいレベルの効率、イノベーション、品質を引き出すことができます。

このセクションでは、以下のベストプラクティスについて説明します。

- [シームレスで end-to-end の統合ツールチェーンの実装](#)
- [DevSecOps 用の end-to-end CI/CD パイプラインの実装](#)
- [コラボレーションツールとプラクティスの採用](#)
- [反復タスクの自動化](#)
- [開発エクスペリエンスを定期的に見直し、反復する](#)
- [効果的なプロジェクト管理プラクティスの採用](#)
- [ナレッジ管理の実装](#)
- [拡張性とカスタマイズの提供](#)
- [オペレーションの最適化](#)
- [データ駆動型インサイトの使用](#)
- [プラットフォームベースのアプローチを採用する](#)

シームレスで end-to-end の統合ツールチェーンの実装

シームレスで end-to-end の統合ツールチェーンを実装することは、生成 AI を活用した開発エクスペリエンスを作成するための基本的なベストプラクティスです。コアアイデアは、ソフトウェアチームが SDLC 全体で利用できるツールとプラットフォームの一貫したエコシステムを確立することです。チームはツールチェーンを使用して、継続的な運用を計画、考案、コード化、構築、テスト、デプロイ、管理できます。生成 AI 機能をこのツールチェーンに統合することで、あらゆる段階で AI 支援を利用できるようにします。この統合により、手動のハンドオフが軽減または排除され、コンテキストの切り替えが軽減され、データとアーティファクトが異なる開発フェーズ間でスムーズに流れるようになります。例えば、統合開発環境 (IDE) からの AI 生成コードスニペットはバージョン管理シ

システムにシームレスに流れ、デプロイプラットフォームからの AI を活用した分析はプロジェクト管理ツールに情報を提供できます。これにより、開発プロセスを改善する継続的なフィードバックループが作成されます。

DevSecOps 用の end-to-end CI/CD パイプラインの実装

この統合されたツールチェーンに基づいて構築するには、DevSecOps end-to-end の継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを実装します。この AI を活用したパイプラインは、ソフトウェア配信プロセスを合理化する重要なコンポーネントです。これにより、新しいアプリケーションや更新をより迅速かつ確実にリリースできます。SDLC 全体にセキュリティプラクティスを埋め込むことで、脆弱性をより早く特定して対処できるため、全体的なコストとリスクを軽減できます。パイプラインには、継続的な統合とテストからセキュリティチェックとデプロイまで、あらゆる段階で AI を組み込む必要があります。たとえば、AI を使用してコードコミットをほぼリアルタイムで分析できるため、潜在的な統合の問題が発生する前に予測できます。CI/CD パイプラインでは、生成 AI を使用して、最新の脅威インテリジェンスに基づいてセキュリティポリシーを自動的に更新することもできます。

コラボレーションツールとプラクティスの採用

開発インフラストラクチャを強化するときは、ヒューマン要素を忘れないでください。ソフトウェア開発は本質的に共同作業です。これには、開発者、デザイナー、製品マネージャー、スクラムマスター、ビジネスアナリスト、その他の利害関係者で構成される部門横断的なチームが含まれます。これらの個人は一人となり、アイデアを生み出します。最新のコラボレーションツールを使用し、オープンなコミュニケーションと知識共有の文化を育むことで、ソフトウェア開発チームの生産性と有効性を大幅に向上させることができます。AI を活用したソフトウェア開発エクスペリエンスでは、これらのツールは新しい側面を取ります。AI をコラボレーションプラットフォームに統合して、チームメンバー間のコミュニケーションと知識共有をより効果的に行うことができます。AI アシスタントは、一般的な質問に答えたり、議論をまとめたり、競合を仲介したりできます。生成 AI は、改善を自動的に提案したり、潜在的な問題を特定したりすることで、コードレビュープロセスを強化できます。さらに、AI を使用して、プロジェクトが進化するにつれてほぼリアルタイムで更新する動的でコンテキスト対応のドキュメントを作成し、すべてのチームメンバーが最新の関連情報にアクセスできるようにすることができます。

反復タスクの自動化

生成 AI を使用して時間のかかる日常的なアクティビティを処理することで、ソフトウェアチームはイノベーションを促進し、ビジネスインパクトをもたらす価値の高いクリエイティブな作業に集中

できるようになります。反復タスクの例としては、定型コードの生成、テストデータの作成、ドキュメントの記述、初期プロジェクト計画のドラフト作成などがあります。これらのタスクを AI にオフロードすることで、チームメンバーはよりクリエイティブで戦略的な作業に集中できます。たとえば、AI を活用したコード補完ツールは、コンテキストとコーディングパターンに基づいて関連するコードスニペットを提案することで、コーディングプロセスを大幅に高速化できます。同様に、生成 AI はコードの変更に応じて技術ドキュメントを自動的に作成および更新できます。これにより、ドキュメントが最新に保たれ、このタスクに通常必要な手動作業が軽減されます。テストでは、AI は要件とコード分析に基づいて包括的なテストケースを生成できるため、テストカバレッジが向上し、エッジケースが見落とされる可能性が低くなります。これらの反復タスクをインテリジェントに自動化することで、生成 AI は開発タイムラインを短縮し、一貫性を向上させ、人為的ミスを減らします。その結果、ソフトウェア出力の品質が向上します。

開発エクスペリエンスを定期的に見直し、反復する

ソフトウェア開発エクスペリエンス自体は、継続的な改良が必要な製品として扱う必要があります。これには、開発ライフサイクル、ツール、プラクティスのあらゆる側面を定期的に見直し、反復するための体系的なプロセスを確立することが含まれます。ツールチェーン、ワークフロー、プロセス全体を定期的に評価します。製品マネージャー、デザイナー、アーキテクト、開発者、テスター、運用担当者など、さまざまな役割のすべてのチームメンバーからフィードバックを収集します。問題点、ボトルネック、強化の機会を特定するように依頼します。たとえば、チームは CI/CD パイプラインのパフォーマンスを四半期ごとにレビューし、ビルド時間、デプロイ頻度、エラー率などのメトリクスを分析して、最適化が必要な領域を特定できます。生成 AI 機能は急速に進化し続けるため、ワークフローをさらに合理化したり、SDLC のすべてのロールで機能を強化したりする可能性のある新しい AI を活用したツールや機能を一貫して評価することが重要です。

効果的なプロジェクト管理プラクティスの採用

複雑なソフトウェア開発作業を効果的に調整するには、AI で強化されたプロジェクト管理プラクティスを採用します。このコンテキストでは、効果的なプロジェクト管理は従来の方法論を超えています。SDLC 全体の計画、実行、モニタリングを強化する AI 拡張アプローチを採用しています。アジャイルフレームワークは柔軟性、コラボレーション、迅速な反復を促進し、生成 AI を使用してこれらのプロセスを最適化できます。例えば、生成 AI は過去のプロジェクトデータを分析してより正確な見積りを行い、ビジネス目標と顧客のフィードバックに基づいてユーザーストーリーを自動的に生成して優先順位付けし、チームのパフォーマンスに関するインテリジェントなインサイトを提供できます。AI を活用したプロジェクト管理ツールは、潜在的な障害を予測し、チームメンバーのスキルとワークロードに基づいて最適なタスク割り当てを提案できます。AI を活用した機能をプロジェ

クト管理プラクティスに統合することで、可視性を高め、データ駆動型の意思決定を迅速に行い、チームメンバーが共通の目標に向けて連携し、効率的に作業できるようになります。

ナレッジ管理の実装

AI を活用したソフトウェア開発エクスペリエンスが成熟したら、堅牢なナレッジ管理システムを実装します。堅牢なナレッジ管理システムは、貴重なインサイト、ベストプラクティス、ソリューションの取得、整理、アクセス許可の付与に役立ちます。SDLC のすべてのチームメンバーは、システムに簡単にアクセスできます。生成 AI を使用して、組織とともに進化する動的でインテリジェントなナレッジベースを作成します。例えば、AI はコードの変更、会話、プロジェクトアーティファクトに基づいてドキュメントを自動的に生成および更新できるため、手動による介入なしに情報が最新のままになります。生成 AI はインテリジェントな検索機能を強化し、チームメンバーが正確な用語を知らなくても、自然言語クエリを使用して関連情報をすばやく見つけるのに役立ちます。さらに、生成 AI は、現在のタスクや課題に基づいて、関連情報をチームメンバーにプロアクティブに提示できます。これは、すべてのロールの意思決定と問題解決を強化する仮想の指導者として機能します。AI を活用したナレッジ管理システムを実装することで、サイロを壊し、オンボーディングを加速し、冗長な作業を減らし、ソフトウェア開発チーム全体で継続的な学習とイノベーションの文化を育むことができます。

拡張性とカスタマイズの提供

ソフトウェア開発における生成 AI の利点を最大化するには、AI を活用したツールとプラットフォームが拡張可能でカスタマイズ可能であることを確認してください。これにより、AI 機能を特定のニーズ、ワークフロー、テクノロジースタックに合わせて調整できます。たとえば、独自のコードベースとドキュメントで AI モデルを微調整したり、特定のタスク用のカスタム AI 搭載ツールを作成したり、AI 機能を既存のツールやプロセスに統合したりできます。この拡張性は、AI を活用した開発エクスペリエンスを進化させ、組織の変化するニーズを満たすのに役立ちます。また、特定のドメインまたはプロジェクトタイプのエクスペリエンスを最適化するのに役立ちます。

オペレーションの最適化

生成 AI は、ソフトウェアオペレーションとメンテナンスの最適化に重要な役割を果たします。AI 機能を運用ツールとプロセスに統合することで、運用に最適化します。例えば、生成 AI を使用して、ログデータをほぼリアルタイムで分析し、潜在的なシステム障害を予測し、定期的なメンテナンスタスクを自動化します。生成 AI は、複雑な分散システム間でイベントを相関させることで、根本原因の分析にも役立ちます。これにより、システムの信頼性が向上し、ダウンタイムが短縮され、運用チームがより戦略的なイニシアチブに集中できるようになります。

データ駆動型インサイトの使用

AI を活用した開発ジャーニー全体で、データ駆動型のインサイトを使用します。SDLC のすべてのステージからデータを収集、分析、処理するシステムを実装します。これには、コードメトリクス、テスト結果、デプロイデータ、ユーザーフィードバック、運用パフォーマンスが含まれます。生成 AI を使用して、人間のオブザーバーには明らかではないパターンやインサイトを発見します。次に、これらのインサイトを開発プロセスにフィードバックし、アーキテクチャ上の決定から機能の優先順位付けまで、あらゆる情報を提供します。

プラットフォームベースのアプローチを採用する

ソフトウェア開発における生成 AI の利点を最大限に活用するには、プラットフォームベースのアプローチを採用します。SDLC のすべての側面に AI 機能を組み込む包括的な統合プラットフォームを作成します。プラットフォームは、一貫したユーザーエクスペリエンス、一元的な管理とデータ、さまざまなツールとプロセス間のシームレスな統合を提供する必要があります。これにより、AI の利点が組織全体で一様に利用可能になり、複数の異なる AI ツールを管理するオーバーヘッドが削減され、AI 機能の継続的な改善と拡張のための基盤が提供されます。

ソフトウェア開発における生成 AI の成功を測定する

生成 AI を活用したソフトウェア開発エクスペリエンスを実装した場合の効果を効果的に測定するには、ソフトウェア開発ライフサイクル (SDLC) のさまざまな側面にまたがる包括的なメトリクスセットを確立する必要があります。これらのメトリクスは、効率と生産性の即時の改善を把握し、ソフトウェア品質、チームの満足度、ビジネス価値の長期的な向上を反映する必要があります。

このセクションで推奨されるメトリクスを効果的に使用するには、以下を実行します。

1. ベースラインを確立する – AI を活用した開発エクスペリエンスの実装を開始する前に、これらのメトリクス全体で現在のパフォーマンスに関する包括的なデータを収集してください。これにより、明確な出発点が提供され、後で意味のある比較を行うのに役立ちます。
2. 現実的なターゲットを設定する – ベースラインを手元に用意し、メトリクスごとに達成可能な改善目標を設定します。野心的でありながら現実的であること。持続可能な進行状況は、多くの場合増分であることに注意してください。
3. 継続的モニタリングの実装 – 自動ツールを使用して、環境内のこれらのメトリクスのデータを絶えず収集および分析します。ほぼリアルタイムのモニタリングは、進捗状況をモニタリングし、問題や機会をすばやく特定するのに役立ちます。
4. 定期的なレビューの実施 – 自分とチームが目標に対する進捗状況を徹底的に評価する四半期または半期のレビューセッションをスケジュールします。これらのセッションを使用して、さらなる改善が必要な分野を特定し、成功を祝います。
5. 反復と調整 — 得られたインサイトに基づいて、生成 AI の実装を継続的に改善し、必要に応じてターゲットを調整します。

このセクションでは、以下のカテゴリのメトリクスについて説明します。

- [デプロイ速度](#)
- [コード品質](#)
- [運用効率](#)
- [チームの生産性と満足度](#)
- [ビジネスへの影響](#)

デプロイ速度

以下のデプロイ速度メトリクスを測定することを検討してください。

メトリクス	説明
市場投入までの時間	アイデアの構想から本番デプロイまでの時間を短縮する
スプリント速度	チームによってスプリントごとに完了したストーリーポイントの増加を追跡する
コードコミット頻度	開発サイクルの加速を示すコードコミットの増加をモニタリングする
プルリクエストの解決時間	リポジトリ内のコード変更を確認およびマージするのにかかる時間の減少を評価する
リリース速度	四半期または年あたりのリリース数の増加を測定する

コード品質

次のコード品質メトリクスを測定することを検討してください。

メトリクス	説明
欠陥密度	ソフトウェアバグの削減を測定する
コードカバレッジ	コードベース全体のテストカバレッジ率の増加を追跡する
技術的負債	特定された技術的負債の経時的な減少をモニタリングする
静的コード分析スコア	自動分析ツールに基づいてコード品質の改善を評価する

運用効率

以下の運用効率メトリクスを測定することを検討してください。

メトリクス	説明
デプロイ頻度	成功したデプロイの数の増加を測定する
平均復旧時間 (MTTR)	システム障害からの復旧にかかる時間の削減を追跡する
変更失敗率	デプロイの失敗につながる変更の割合の減少をモニタリングする

チームの生産性と満足度

以下のチームの生産性と満足度のメトリクスを測定することを検討してください。

メトリクス	説明
生産性の向上	各タスクの生産性の向上率をモニタリングする
満足度スコア	定期的なアンケートを実施して、チームの士気と仕事への満足度の向上を評価します。
ナレッジ共有の効率	チームが情報の検索や反復的な質問に費やす時間の削減を測定する
オンボーディング時間	新しいチームメンバーが生産性を上げるために必要な時間の減少を追跡する

ビジネスへの影響

以下のビジネスへの影響メトリクスを測定することを検討してください。

メトリクス	説明
機能の採用率	リリースした新機能によるユーザーエンゲージメントの増加を測定する

メトリクス	説明
顧客満足度スコア	ユーザーのフィードバックと評価の改善を追跡する
収益への影響 (直接的および間接的)	リリース速度の向上または生産性の向上に起因する収益の増加を評価する

結論

この戦略ドキュメントでは、生成 AI を活用したソフトウェア開発エクスペリエンスの概要を説明します。[55-1フレームワークの 5](#) つのディメンションである、調査、統合、対話、反復、影響について説明します。これらのディメンションは、ソフトウェア開発ライフサイクル (SDLC) 全体で生成 AI を統合するための戦略的ロードマップを提供します。また、このフレームワークを正常に実装するために必要な[基本的な機能](#)についても説明します。機能は、プロジェクト管理、DevSecOps、AI アシスタント、ナレッジ管理などの分野にまたがります。生成 AI を統合する際に考慮すべき[ベストプラクティス](#)を提供し、[メトリクス](#)を使用して生成 AI がソフトウェア開発エクスペリエンスに与える影響を測定するのに役立ちます。

生成 AI をソフトウェア開発プロセスに統合することは、イノベーションを加速し、品質を向上させ、生産性を高める可能性のあるパラダイムシフトを表しています。ただし、これは 1 回限りの実装ではないことを認識することが重要です。これは、持続的な労力と継続的な改良を必要とする継続的な進化です。

このジャーニーを開始するときは、まず組織の現在の機能と準備状況を徹底的に評価することをお勧めします。[AWS 評価ツール](#)は AI を活用したソフトウェア開発評価ツールで、優先度の高い分野を特定し、カスタマイズされた実装ロードマップを作成するのに役立ちます。

リソース

主要な優先順位領域を特定したら、以下のリソースがロードマップの実装に役立ちます。

AWS ドキュメント

- [Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する](#) (AWS 規範ガイダンス)
- [インラインコード生成とアシスタントコード生成のための Amazon Q Developer のベストプラクティス](#) (AWS 規範ガイダンス)
- [Amazon Bedrock エージェントとナレッジベースを使用して、完全に自動化されたチャットベースのアシスタントを開発する](#) (AWS 規範ガイダンス)
- [生成 AI AWS を使用したでのアプリケーション開発およびメンテナンス運用モデルの変換](#) (AWS 規範ガイダンス)
- [Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める](#) (AWS 規範ガイダンス)

AWS ブログ投稿とチュートリアル

- [Amazon Q ブログ記事](#)
- [Amazon Q でソフトウェア開発ライフサイクルを加速する](#) (AWS ブログ記事)
- [ソリューションアーキテクト AI エージェントの構築 AWS : Amazon Bedrock を活用してアーキテクチャとデプロイを自動化する](#) (AWS ビデオ)
- [生成 AI を活用したテクノロジーオペレーション](#) (AWS ブログ記事)
- [Amazon Q Developer を使用して Java アプリケーションをモダナイズする](#) (AWS ブログ記事)
- [Amazon Bedrock を使用してソフトウェア開発パイプラインのコードを生成、評価、理解する](#) (AWS ブログ記事)

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、[RSS フィード](#) をサブスクライブできます。

変更	説明	日付
初版発行	該当しない	2025 年 4 月 18 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。