# 再帰的知性の進化：DeNAのAI駆動開発エコシステムにおける戦略的分析と実践 (2026年版)

**Gemini 3 pro**

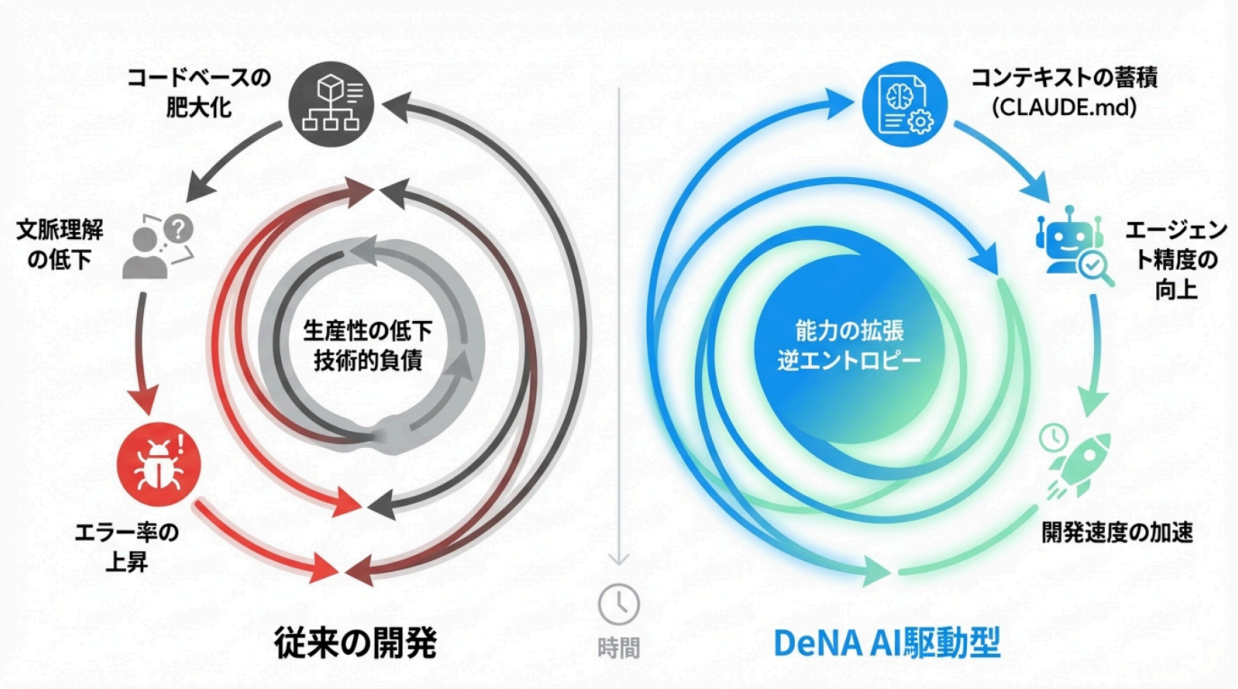
## エグゼクティブサマリー

2026年1月現在、ソフトウェアエンジニアリングの地平は根本的な相転移を遂げました。私たちは、チャットボットやオートコンプリートの散発的な利用に特徴づけられた「AI支援（AI-Assisted）」の時代を超え、「AI駆動（AI-Driven）」開発の時代へと突入しました。この新しいパラダイムにおいて、人工知能は単なる道具ではなく、コード、ドキュメンテーション、品質保証のライフサイクル全体を統治するシステム的なインフラストラクチャとして機能しています。本レポートは、株式会社ディー・エヌ・エー（DeNA）の永田浩矢氏による先駆的な方法論「育てるほど楽になる AI 開発体制を作っている話」（2026年1月6日公開）を起点とし、この転換を包括的に分析するものです[1]。

分析の結果、DeNAの戦略はエンタープライズAI導入における決定的な成熟段階を示していることが明らかになりました。彼らは、個々の生産性向上という局所的な視点から、システム的な「コンテキスト・エンジニアリング」へと焦点を移すことで、2024年から2025年にかけて業界を悩ませた主要なボトルネックである「コンテキスト・ギャップ（文脈の欠如）」を解消しました。Model Context Protocol (MCP) の戦略的展開、Claude Codeを用いた高度なエージェンティック・ワークフロー、そして「AI All-In」宣言に基づく抜本的な組織再編を通じて、DeNAは時間の経過とともに複雑性のコストが減少していく開発環境を構築しました。本レポートでは、この現象を**「再帰的コンテキスト最適化（Recursive Contextual Optimization）」**と定義します。

技術アーキテクチャ、組織心理学、そして2030年を見据えた未来予測にまたがる本レポートは、2026年の「DeNAモデル」が現代のソフトウェア企業にとっての青写真となると論じます。ここでは、AIを「育てる」ための技術的メカニズム、ガバナンスにおけるSWET（Software Engineering in Test）の役割、そして人間のエンジニアが「コードの書き手」から「知性の設計者（Context Architect）」へと変貌するエージェント型労働力への移行がもたらす広範な影響について詳述します。

## 再帰的進化：従来の技術的負債とDeNAの「育てるAI」モデルの比較




左側は、時間の経過とともに複雑さが増し、生産性が低下する従来のソフトウェア開発サイクルを示しています。右側は、DeNAのAI駆動型モデルであり、プロジェクトのコンテキスト（CLAUDE.md、MCPサーバー）が蓄積されるにつれて、AIエージェントの推論精度が向上し、開発者の認知負荷が軽減される「逆エントロピー」のサイクルを描いています。

---

# 第1章：パラダイムシフト ー 支援から自律へ

## 1.1 2026年におけるAI駆動開発（ADD）の定義

DeNAの取り組みの重要性を理解するためには、まず2026年初頭における業界の状況を文脈化する必要があります。2023年から2025年にかけての期間は「AI支援（AI-Assisted）」開発の時代として特徴づけられました。このフェーズでは、GitHub Copilotや初期のChatGPTなどのツールは「賢いタイプライター」として機能していました。人間が唯一のドライバーであり、特定のタスク（例：「JSONをパースする関数を書いて」「このエラーを説明して」）のためにAIを明示的に呼び出す必要がありました[2]。

しかし、永田浩矢氏が記述し、DeNAで実践されている「AI駆動（AI-Driven）」モデルは、制御の根本的な逆転を表しています[1]。このパラダイムにおいて、AIシステムは開発ライフサイクルの中に常駐する半自律的なエージェントとして機能します。AIは単にプロンプトを待つだけでなく、リポジトリを能

動的に監視し、リファクタリングを提案し、ドキュメントを更新し、自己修正ループを実行します。

DeNAの「AI駆動」アプローチの決定的な特徴は、**コンテキストの永続性（Context Persistence）**にあります。支援型モデルでは、チャットセッションが終了した瞬間にコンテキストは失われていました（エフェメラルな文脈）。対してDeNAの2026年モデルでは、プロジェクトの「知識」——コーディング規約、アーキテクチャの決定事項、レガシーな制約事項など——が、AIが継続的に更新・参照可能な形式（CLAUDE.mdやベクトルストアなど）にシリアライズされています [5]。これにより、成長するほどに効率化される開発環境、すなわちプロジェクトを「学習」するシステムが実現されるのです。

## 1.2 「育てる」というメタファー：ジュニアエンジニアとしてのAI

永田氏のブログタイトル「育てるほど楽になる AI 開発体制」は、2026年時代のエンジニアリングを理解する上で不可欠な生物学的メタファーを活用しています。このシステムでは、AIは静的なツールとしてではなく、**「育成対象のジュニアエンジニア」**として扱われます。

- オンボーディング（学習）：人間のエンジニアがREADMEやCONTRIBUTING.mdを読むのと同様に、AIエージェントはリポジトリのコンテキストファイルを読み込みます [5]。
- 修正と指導：AIが間違いを犯した場合（例：非推奨のライブラリを使用するなど）、その修正は単にコードに適用されるだけでなく、システムの指示書（コンテキストファイル）にも記録されます [1]。
- 成長：時間の経過とともに、AIの「メンタルモデル」はシニアエンジニアの基準と一致していき、詳細なプロンプトなしでも適切な出力を生成できるようになります。

このシフトは、初期のAI導入において顕著であった「レビュアー疲労（Reviewer Fatigue）」の問題に対処するものです [1]。以前は、人間がAIの生成した粗悪なコードを修正するために、自分で書く以上の時間を費やしていました。システムを「育てる」ことで、DeNAはAIの出力が人間の介入を最小限に抑えられるティッピングポイント（転換点）に到達することを目指しています。

## 1.3 従来の自動化との決定的な違い

従来のCI/CDや静的解析ツール（Linter）と、このAI駆動体制の違いは「柔軟性」と「文脈理解」にあります。従来のツールは決定論的であり、予め定義されたルール（「行末にセミコロンがない」など）には強いものの、「この変更はビジネスロジック的に既存の決済フローと矛盾しないか？」といった文脈的判断は不可能でした。

DeNAの体制は、大規模言語モデル（LLM）の推論能力を開発パイプラインに組み込むことで、この壁を突破しています。具体的には、GitHub Actions上で動作するClaude Codeのエージェントが、コードの変更意図を理解し、関連するドキュメントを探し出し、整合性をチェックするという、従来人間にしかできなかった認知タスクを自動化しています [7]。これは「自動化（Automation）」から「自律化（Autonomy）」への進化と言えます。

---

# 第2章：DeNAの戦略的「AI All-In」アーキテクチャ

## 2.1 組織的マンデートと南場会長のビジョン

DeNAの技術的成果は、極めて野心的な企業戦略によって支えられています。南場智子会長のリーダーシップの下、DeNAは「AI All-In」という方針を宣言しました [9]。これは単なるスローガンではなく、「既存事業の人員をAIで半減させ、残りの人材を新規事業に振り向ける」という、リソース配分を伴う具体的かつ過激な経営目標です [11]。

このトップダウンの指令は、以下の2つの主要な実行部隊へと展開されています。

1. **AIシステム部 & AI戦略推進室:** コアとなるインフラストラクチャの構築とR&Dを担当します。ここでは、先端AI技術の研究開発と事業貢献の両立を目指し、認証基盤の内製化やデータ基盤の整備を進めています [13]。
2. **SWET (Software Engineering in Test):** 従来はテスト自動化や品質保証に焦点を当てていたこのグループは、AI時代の「開発者体験（Developer Experience）」と「AI品質」の守護者へと進化しました [15]。SWETは、AIツールが生成するコードの品質管理や、開発プロセスへのAI統合におけるガバナンスを担っています。

## 2.2 「100の活用事例」に見る自律性の階層

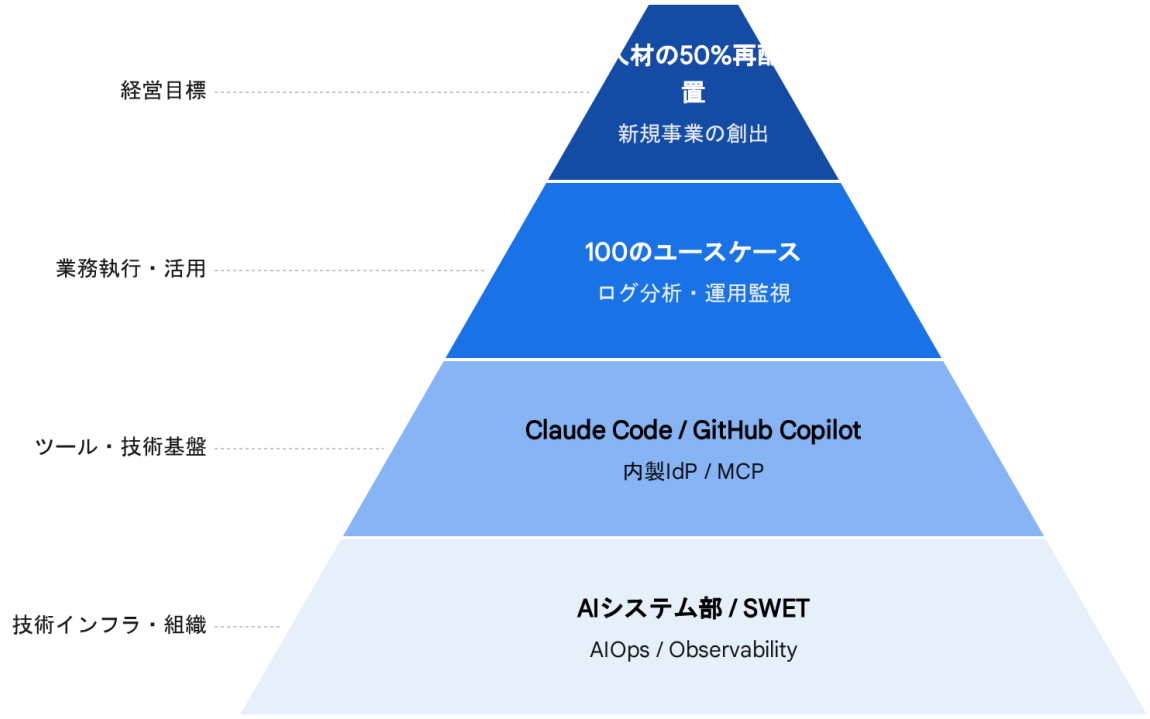
戦略の具体化として、DeNAは「現場のAI活用事例100選」を体系化し、公開しました [17]。この取り組みは、社内での知識共有（「どやる時間」[20]）と外部への採用ブランディングの両面で機能しています。

これらの事例はランダムな集合ではなく、自律性のレベルに応じたタクソノミー（分類体系）に従っています。

- レベル**1**：情報検索・整理**:** 技術情報のキャッチアップ、議事録の要約など、情報の入力と整理を効率化するフェーズ [17]。
- レベル**2**：生成・プロトタイピング**:** コード生成、デザイン案の作成、動画生成など、クリエイティブな出力を補助するフェーズ [21]。
- レベル**3**：運用分析・監視**:** ログ分析、異常検知、RCA（根本原因分析）の自動化など、システムの運用を支えるフェーズ [21]。
- レベル**4**：体系的自律化（**Systematic Automation**）**:** 永田氏が記述する「育てる開発体制」のように、AIが自律的にコンテキストを維持・更新し、開発サイクル自体を回すフェーズ [1]。

この構造化されたアプローチにより、DeNAはAI導入の「幻滅期」を回避し、ROI（投資対効果）が測定可能な産業レベルの実装へと移行することに成功しています。

## DeNAの「AI All-In」戦略的階層構造




この図は、DeNAのAI戦略の構造を示しています。基盤となるのは「AIシステム部」と「SWET」による技術インフラであり、その上にMCPやClaude Codeなどの「ツール群」が位置し、最上位には南場智子会長が掲げる「人材の再配置と新規事業の創出」という経営目標があります。

Data sources: DeNA Engineering, SB BIT, SpinFlow, Workstyle Evolution, DeNA CSR

---

# 第3章：技術的コア ― AIを「育てる」メカニズム

## 3.1 汎用AIエージェントの失敗とコンテキストの壁

永田氏の洞察は、2024年から2025年にかけて多くの企業が直面した「汎用PRエージェント」の失敗体験に基づいています。初期のAIによるプルリクエスト（PR）レビューの試みは、しばしば表層的なコメント（「変数名をもっと説明的に」「コメントを追加して」など）に終始していました [1]。

これらのエージェントには**「プロジェクト固有のコンテキスト」**が欠落していました。彼らは、プロジェクトXが特定のエラーハンドリングラッパーを使用していることや、チームYが関数型プログラミングのパターンを好むといった「暗黙知」を持っていません。その結果、AIの指摘は一般的なプログラミ

ングの教科書としては正しくても、そのプロジェクトの文脈においては「ノイズ」となり、開発者の認知負荷を増大させていました。これは、AI導入が逆に生産性を下げる「幻滅の谷」の典型的な原因でした。

## 3.2 ソリューション：再帰的コンテキスト注入 (Recursive Context Injection)

永田氏の提唱する「AI駆動」アプローチの革新性は、コンテキストの注入を体系化した点にあります。これは、**Model Context Protocol (MCP)** と **Living Documentation**（生きたドキュメント）の組み合わせによって達成されています。

### 3.2.1 CLAUDE.md と context.md の役割

「育てる」プロセスの中心には、しばしば CLAUDE.md や context.md と名付けられるファイルが存在します [5]。これらは人間向けの読み物としてのドキュメントではなく、コード化されたシステムプロンプトです。

- 構造と内容: これらのファイルには、以下のようなプロジェクトの「掟」が簡潔に記述されています。
  - ビルドコマンド (npm run build) やテスト手順。
  - アーキテクチャ上の制約（例：「DBアクセスは必ずService層を経由すること」）。
  - スタイルガイド（例：「let よりも const を優先する」）。
  - レガシーコードの扱い（例：「moment.jsは非推奨。新規コードではdate-fnsを使用せよ」）。
- フィードバックループの永続化: 開発者がAIの出力を修正した際、その修正は単にコードに反映されるだけではありません。開発者（あるいはメモリエージェント自体）が CLAUDE.md を更新します。
  - 具体例: AIが非推奨のライブラリを提案した場合、開発者は CLAUDE.md に「禁止事項」を追加します。
  - 結果: 次回以降、どの開発者が**AI**を使用しても、AIはこのルールを遵守します。これにより、「チームの脳」が強化され、同じ指摘を繰り返す必要がなくなります。これが「育てるほど楽になる」メカニズムの正体です [1]。

## 3.3 インフラストラクチャ：Claude Code と MCP の統合

DeNAの実装は、Anthropic社の高度なCLIツールである **Claude Code** [22] と、**Model Context Protocol (MCP)** [24] の統合の上に成り立っています。

**MCP (Model Context Protocol)** は、2024年後半にAnthropicによって導入され、2025年にはOpenAIやGoogleにも採用された「AIのためのUSB-C」とも呼ぶべき標準規格です [25]。MCP以前、LLMをデータベースやGitHubリポジトリに接続するには、個別のAPI統合が必要でした。しかしMCPの導入により、DeNAのエンジニアは以下のような情報をAIエージェントに安全かつ標準的な方法で公開する「MCPサーバー」を立ち上げることができます。

- **PostgreSQL**スキーマ: AIがDB構造を理解し、正しいSQLを発行できるようにする [25]。
- **Git**履歴: AIが「なぜ6ヶ月前にこの行が変更されたのか」という経緯を理解できるようにする [27]。
- 社内ナレッジベース: SlackのログやNotionのドキュメントを検索可能にする [25]。

MCPによって、DeNAのAIエージェントは、単にエディタ上のテキストを見るだけでなく、開発環境全体の「状態」を把握できるようになりました。この可視性こそが、AIを「単なるテキスト予測器」から「文脈を理解する開発者」へと変貌させる鍵です。

---

# 第4章：ワークフロー詳解 ー 自己修正型パイプラインの実装

## 4.1 自動ドキュメント更新パイプライン

「育てる」哲学の最も強力な適用例の一つが、自己更新型ドキュメンテーション・パイプラインです。

従来の開発では、ドキュメントは常にコードの後塵を拝していました。コードが変更された後に人間が手動で更新するため、常に陳腐化のリスクがありました。DeNAのAI駆動モデルでは、プルリクエスト（PR）をトリガーとしたGitHub Actionsによって、ドキュメントの更新が自動化されています [7]。

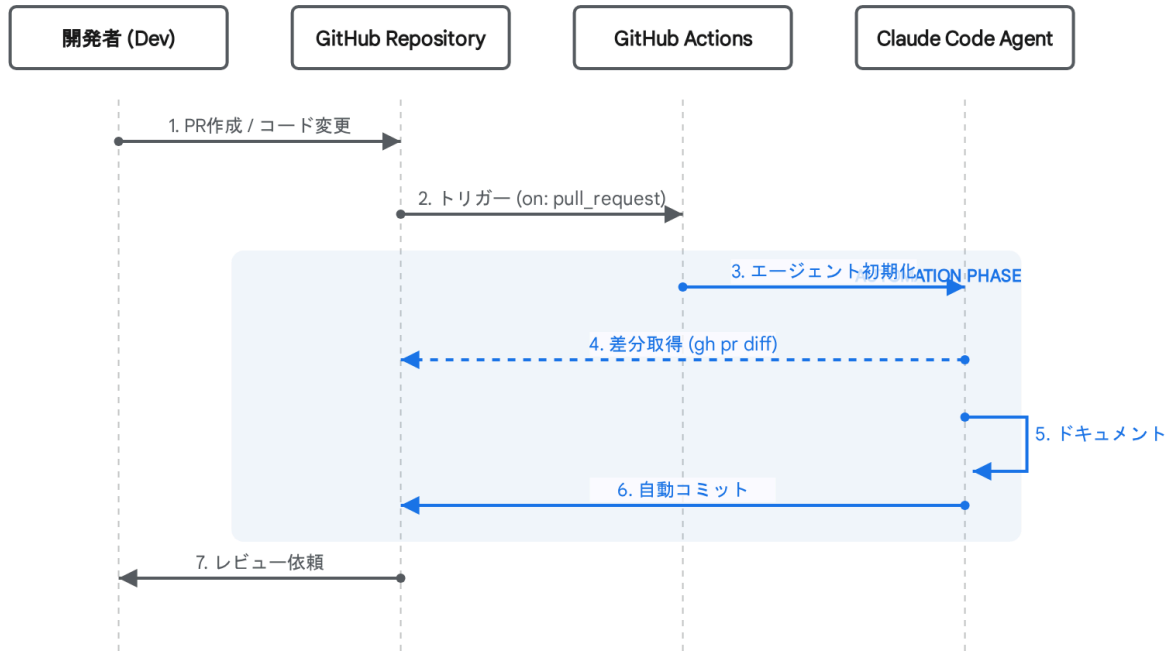
ワークフローの詳細 [7]:

1. トリガー: 開発者がコード変更を含むPRを作成または更新する。
2. 分析: CI上で動作するAIエージェント（claude-code-action）が git diff を分析し、変更内容を理解する。
3. 検知: 例えば auth.ts の関数シグネチャが変更された場合、AIはそれに関連するドキュメント（docs/auth.md や JSDocコメント）を特定する。
4. アクション: AIはドキュメントの修正案を生成し、同じPR内に新しいコミットとしてプッシュする。
5. レビュー: 人間の開発者は、コードの変更と同時に、AIが更新したドキュメントをレビューする。

このプロセスにより、AIの燃料となる「プロジェクト・コンテキスト」が常に最新かつ正確に保たれます。これは、「より良いコンテキスト → より良いAIコード生成 → より正確なドキュメント → さらに良いコンテキスト」という正のフィードバックループ（好循環）を生み出します。

# 自己修正型ドキュメンテーション・パイプラインのシーケンス図

自動更新ワークフロー




この図は、GitHub ActionsとClaude Codeを利用したドキュメント自動更新フローを示しています。コードの変更がトリガーとなり、AIエージェントが影響範囲を分析し、ドキュメントの修正案をPRに直接コミットするまでの流れを可視化しています。

Data sources: Automate your Documentation, Integrate Claude Code, Automated Documentation (Docusaurus)

## 4.2 評価駆動デリバリー (Evaluation-Driven Delivery: EDD)

AI駆動開発への移行は、**評価駆動デリバリー（EDD）**と呼ばれる新しい方法論を生み出しました[28]。テスト駆動開発（TDD）が「テストコードを先に書く」ことで要件を明確化するように、EDDは「評価基準（ゴール）」を最初に定義します。

DeNAのコンテキストにおけるEDDのプロセスは以下の通りです：

1. ゴール定義: 人間が自然言語でゴールを定義する（例：「APIクライアントにリトライ機構を追加し、指数バックオフを実装せよ」）。
2. コンテキスト収集: AI（MCPクライアント）が関連するファイル、過去の類似PR、CLAUDE.mdの規約を収集する。
3. 計画立案: AIが実装プランを提案する。

4. 人間によるレビュー（ドライバー）: 人間がプランの妥当性を検証する。ここでの人間の役割はコードを書くことではなく、設計の承認です。
5. 実行と評価: AIがコードを実装し、テストを実行してゴールに対する達成度を評価する。

このプロセスにおいて、人間の役割は「コードライター」から「プランレビューア」および「コンテキストキュレーター」へとシフトします。AIがタスクを実行する間、人間はより抽象度の高い「ゴール設定」と「品質基準の策定」に集中できるようになります。

---

# 第5章：品質保証の変革 ― ループの中のSWET

## 5.1 品質管理の再定義

DeNAのSWET（Software Engineering in Test）チームは、この変革において極めて重要な役割を果たしています [16]。AI駆動の世界では、「品質」の定義が拡張されます。それは単に「バグのないコード」であることだけでなく、「AIが理解・保守しやすいコード」であることも含みます。

SWETの新たなマンデート（使命）には以下が含まれます：

- **AIパフォーマンス指標の策定:** AI生成コードの採用率、修正率、レビュー通過率などの定量的指標を定義し、計測する [29]。
- ガードレールの実装: AIがセキュリティ上の脆弱性（幻覚による架空のパッケージのインポートなど）やライセンス違反を引き起こさないよう、CIパイプラインに自動チェック機構（ガードレール）を組み込む [31]。
- テストデータ管理: AIエージェントの性能を評価するための「ゴールデンデータセット（正解データ）」を整備・管理する [32]。

## 5.2 「どやる時間」による集合知の形成

DeNAの品質戦略におけるユニークな文化的側面が「どやる時間」です [20]。これは、各チームが自身のAI活用成功事例を自慢（どやる）し合うために設けられた指定時間です。

一見インフォーマルな活動に見えますが、このメカニズムは組織にとって**「分散型遺伝的アルゴリズム」**として機能しています。あるチームで成功したプロンプトのパターンや、効率的なMCPサーバーの設定は、この場を通じて急速に全社へ伝播します。逆に、効果のなかったアプローチは淘汰されます。このソーシャル・エンジニアリングは、ソフトウェア・エンジニアリングと同様に、全社的なAIリテラシーの底上げに不可欠な要素となっています。

---

# 第6章：人的要素 ― スキルセットとリソース再配置

## 6.1 「50%再配置」の現実と人材要件

南場会長が掲げる「既存事業の人員を半減させる」というビジョン [11] は、AIによる生産性向上が2倍近くに達することを前提としています。この現実は、現場のエンジニアに対して根本的なスキルセットの変化を要求します。

- ジュニアエンジニアの変容: 従来、若手エンジニアはボイラープレート（定型）コードを書くことで経験を積んでいました。しかし、その領域はAIが最も得意とする部分です。これからのジュニアエンジニアには、AIが生成したコードを読み解き、批判的にレビューする能力が求められます。彼らのメンターの一部は、シニアエンジニアの知恵をコンテキストとして保持する「育てられたAI」が担うことになります [21]。
- シニアエンジニアの変容: 彼らの役割は「コンテキスト・アーキテクト」へとシフトします。彼らの価値はもはやコーディングの速度ではなく、複雑な制約条件を CLAUDE.md に言語化し、AIの自律性の境界線を定義する能力にあります。

## 6.2 新たなコアコンピテンシー：コンテキスト・エンジニアリング

2026年のDeNAにおいて、コンテキスト・エンジニアリング（**Context Engineering**）は必須スキルとなっています [34]。これはプロンプト・エンジニアリングとは明確に区別されます。

- プロンプト・エンジニアリングは戦術的です（例：「Xのためのクエリを書いて」）。
- コンテキスト・エンジニアリングは戦略的です（例：「AIが問われずとも自動的にXのクエリを書けるように、ディレクトリ構造とメタデータを設計する」）。

DeNAのエンジニアたちは、AIシステムの「庭師」として、不要なコンテキスト（ノイズ）を剪定し、新しい知識を接ぎ木する作業に従事していると言えます。

---

# 第7章：将来展望 ― 2030年へのロードマップ

## 7.1 エージェンティック・ワークフローとマルチエージェント・オーケストレーション

2026年以降、トレンドは不可避的にマルチエージェント・オーケストレーションへと向かっています 35。
DeNAの開発体制は、現在の単一の「ペアプログラマー」モデルから、専門化されたエージェントによる「分隊（Squad）」モデルへと進化すると予測されます。

- エージェント**A**（アーキテクト）: インターフェースとデータ構造を設計する。
- エージェント**B**（コーダー）: ロジックを実装する。
- エージェント**C**（テスター）: テストケースを作成し、エッジケースを探索する。
- エージェント**D**（レビュアー）: CLAUDE.md のガイドラインに照らしてコードを批判する。

「Serena」のような高度なMCPサーバー [37] は、意味理解とシンボリック検索を可能にし、これらの専門化された分隊メンバーの前身となる技術です。

## 7.2 エンタープライズデータの「USB-C」化

MCPの採用は加速し、すべての社内ツールの標準プロトコルとなると予想されます。2027年までに

は、「MCP準拠（MCP Compliance）」が社内ソフトウェア調達の必須要件となるでしょう[24]。MCP経由で自身の状態をAIエージェントに公開できないツールは、「AIと対話できない」孤立したシステムとして淘汰されていく運命にあります。

---

## 結論

2026年初頭にDeNAで進行している変革は、世界のソフトウェア産業にとっての先行指標（ベルウェザー）としての役割を果たしています。「AIを使う」段階から「AIを育てる」段階へと移行することで、DeNAは持続可能なスケーラビリティの鍵を解き明かしました。彼らは、コードの真の価値はテキストファイルそのものではなく、それを包み込む**コンテキスト（文脈）**にあることを見抜いています。

「AI駆動」開発モデルは、人間を置き換えるものではなく、人間をより高い抽象度のレベルへと引き上げるものです。DeNAのエンジニアたちは、もはや単なるレンガ積み職人ではなく、自己組織化するシステムの設計者です。システムが「学習」し「成長」するにつれて、開発の摩擦は蒸発し、人間の意図とデジタルな実現との間にある純粋な導管だけが残るのです。

業界全体への教訓は明白です。開発者のためにツールを作るのをやめ、開発者がツールを育成できる環境を作り始めるべきです。

# ソフトウェア開発パラダイムの進化ロードマップ (2024-2030)


2024年の「AIアシスト」から、現在の「AI駆動（コンテキスト指向）」、そして2027年以降の「自律的エージェントオーケストレーション」へと至る進化の過程を示しています。DeNAは現在、第2フェーズ（AI Driven）の成熟期に位置しています。

Data sources: DeNA Tech (Docswell), DeNA Engineering Blog, The New Stack, Machine Learning Mastery, arXiv

## 引用文献


1. 育てるほど楽になる AI 開発体制を作っている話 - DeNA Engineering, 1月 12, 2026にアクセス、https://engineering.dena.com/blog/2026/01/ai-driven-develop/
2. The Role of AI in Enhancing Software Development Productivity and ..., 1月 12, 2026にアクセス、https://www.allata.com/insights/the-role-of-ai-in-enhancing-software-development-productivity-and-collaboration/
3. Unlocking the value of AI in software development - McKinsey, 1月 12, 2026にアクセス、

https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/unlocking-the-value-of-ai-in-software-development
4. AI-Driven Development (ADD) — How It's Revolutionizing the Way ..., 1月 12, 2026にアクセス、https://medium.com/@qapournima/ai-driven-development-add-how-its-revolutionizing-the-way-we-build-technology-c515d522b033
5. Claude Code: Best practices for agentic coding - Anthropic, 1月 12, 2026にアクセス、https://www.anthropic.com/engineering/claude-code-best-practices
6. A week with Claude Code: lessons, surprises and smarter workflows, 1月 12, 2026にアクセス、https://dev.to/ujjavala/a-week-with-claude-code-lessons-surprises-and-smarter-workflows-23ip
7. Automate Your Documentation with Claude Code & GitHub Actions, 1月 12, 2026にアクセス、https://medium.com/@fra.bernhardt/automate-your-documentation-with-claude-code-github-actions-a-step-by-step-guide-2be2d315ed45
8. Automated Documentation with Claude Code: Building Self ..., 1月 12, 2026にアクセス、https://medium.com/@dan.avila7/automated-documentation-with-claude-code-building-self-updating-docs-using-docusaurus-agent-2c85d3ec0e19
9. 【2025年振り返り】「AIオールイン」から始まった2025年。DeNA ..., 1月 12, 2026にアクセス、https://engineering.dena.com/blog/2025/12/tech-pr_summary/
10. DeNAがAIに全賭けする理由とは？成功企業の最新AI戦略と未来予測, 1月 12, 2026にアクセス、https://note.com/bekonjapan/n/n2b04ee67e33b
11. DeNA南場智子氏「7つの AI活用術」と企業活用提案, 1月 12, 2026にアクセス、https://www.spinflow.jp/news/DeNA%E5%8D%97%E5%A0%B4%E6%99%BA%E5%AD%90%E6%B0%8F%E3%80%8C7%E3%81%A4%E3%81%AEAI%E6%B4%BB%E7%94%A8%E8%A1%93%E3%80%8D%E3%81%A8%E4%BC%81%E6%A5%AD%E6%B4%BB%E7%94%A8%E6%8F%90%E6%A1%88
12. 【DeNA】南場智子会長の思い切ったAIオールイン！宣言 - note, 1月 12, 2026にアクセス、https://note.com/yoshiyuki_hongoh/n/nb3556c9e0aa4
13. DeNAが見据える「AIカンパニー」への道筋とその体制 - ビジネス+IT, 1月 12, 2026にアクセス、https://www.sbbit.jp/article/cont1/35547
14. DeNA TechCon2018 研究開発と事業貢献を両立させるAI組織の ..., 1月 12, 2026にアクセス、https://speakerdeck.com/swordheart/dena-techcon2018-ai-organization
15. Making AI easy for software developers - DENA, 1月 12, 2026にアクセス、https://www.designing-electronics.com/making-ai-easy-for-software-developers-2/
16. AIと開発プロセスの改善チャレンジ | BLOG - DeNA Engineering, 1月 12, 2026にアクセス、https://engineering.dena.com/blog/2025/10/swet-ai-journey/
17. DeNAのAI活用100本ノック全解析！企業のAI導入を成功させる体系 ..., 1月 12, 2026にアクセス、https://gai.workstyle-evolution.co.jp/2025/12/31/dena-ai-100-use-cases-systematic-classification-enterprise-implementation-guide/
18. DeNAが「現場のAI活用事例」100選を全スライド公開ほか - note, 1月 12, 2026にアク

セス、https://note.com/no_ai_no_life/n/n43b5e733c62c
19. プロンプト集を超えて成果に直結する！DeNA流のビジネス特化型 ..., 1月 12, 2026にアクセス、https://tenbin.ai/media/generative_ai/dena-ai-100-business-tips
20. DeNA南場流「人×組織×AI」の未来像, 1月 12, 2026にアクセス、https://wa2.ai/ai-news/dena-namba-jin-soshiki-ai-mirai
21. Technology & Monozukuri | DeNA Co., Ltd. - DeNA Sustainability, 1月 12, 2026にアクセス、https://csr.dena.com/intl/technology/
22. Claude Code GitHub Actions, 1月 12, 2026にアクセス、https://code.claude.com/docs/en/github-actions
23. Claude Code overview - Claude Code Docs, 1月 12, 2026にアクセス、https://code.claude.com/docs/en/overview
24. Model Context Protocol - Wikipedia, 1月 12, 2026にアクセス、https://en.wikipedia.org/wiki/Model_Context_Protocol
25. Introducing the Model Context Protocol - Anthropic, 1月 12, 2026にアクセス、https://www.anthropic.com/news/model-context-protocol
26. Why the Model Context Protocol Won - The New Stack, 1月 12, 2026にアクセス、https://thenewstack.io/why-the-model-context-protocol-won/
27. Claude Code Studio | MCP Servers - LobeHub, 1月 12, 2026にアクセス、https://lobehub.com/mcp/arnaldo-delisio-claude-code-studio
28. From Task-Driven AI Copilots to Goal-Driven AI Pair Programmers, 1月 12, 2026にアクセス、https://arxiv.org/pdf/2404.10225
29. Five Metrics That Matter When Using AI in Test and Validation - NI, 1月 12, 2026にアクセス、https://www.ni.com/en/perspectives/five-metrics-when-using-ai-test-validation.html
30. AI Metrics Tracking: Key Insights & Best Practices - Scout, 1月 12, 2026にアクセス、https://www.scoutos.com/blog/ai-metrics-tracking-key-insights-and-best-practices
31. 日本の大企業におけるコード生成AI導入の課題と対策 - ZenTech, 1月 12, 2026にアクセス、https://www.zen-tech.co.jp/blog/code-generation-ai-adoption-japanese-enterprises
32. Two Indispensable Tools for Measuring the Quality of AI Systems, 1月 12, 2026にアクセス、https://odsc.medium.com/two-indispensable-tools-for-measuring-the-quality-of-ai-systems-4db5318ae689
33. なぜ僕らはピボットを恐れなかったのか。AIオールインで乗り越えた ..., 1月 12, 2026にアクセス、https://engineering.dena.com/blog/2025/12/frica-shinsotsu-blog/
34. The Model Context Protocol's impact on 2025 | Thoughtworks Spain, 1月 12, 2026にアクセス、https://www.thoughtworks.com/en-es/insights/blog/generative-ai/model-context-protocol-mcp-impact-2025
35. 7 Agentic AI Trends to Watch in 2026 - MachineLearningMastery.com, 1月 12, 2026にアクセス、https://machinelearningmastery.com/7-agentic-ai-trends-to-watch-in-2026/

36. The state of agentic AI in 2025: what's working, what isn't, and what's ..., 1月 12, 2026にアクセス、https://www.openocean.vc/articles/the-state-of-agentic-ai-in-2025-whats-working-what-isnt-and-whats-next
37. A Deep Dive into the VS Code MCP Server by Juehang Qin, 1月 12, 2026にアクセス、https://skywork.ai/skypage/en/ai-development-vs-code/1979063844504981504
38. Fully Featured AI Coding Agent as MCP Server : r/ClaudeAI - Reddit, 1月 12, 2026にアクセス、https://www.reddit.com/r/ClaudeAI/comments/1jpavtm/fully_featured_ai_coding_agent_as_mcp_server/