Vibe Codingによるソフトウェアエンジニアリングのレベルアップ

発表者紹介

ジョン·ドヒョン

- ROBOCO.IO 主席コンサルタント
- 1995年12月、ユニテル事業部でフロントエンド開発者としてキャリアをスタート
- AWSシニアデベロッパーおよびトレーナーとして8年半在職
- 日本で14年間、ITコンサルタントおよびSI開発者として在職

Vibe Codingとは？

- **AI支援開発** – 自然言語の指示でコードを実装する新しい開発スタイル（ Karpathy, 2025 ）
- **LLMとのペアプログラミング** – 開発者とAIの対話型ソフトウェア開発
- **「感覚でコーディング」**– 詳細な実装はAIが担当し、自然言語で完成形を定義することに集中

なぜVibe Codingなのか？ - 圧倒的な生産性

- **高速開発サイクル** – アイデア → プロトタイプを即時実装、MVP開発を加速
- **問題解決に集中** – 実装よりも要件定義とソリューション設計に集中
- **迅速な反復実験** – 低コストでのアイデア検証、短いフィードバックサイクル

なぜVibe Codingの生産性は高いのか？

- **少人数での大規模開発が可能に** - 幾何級数的に増加するコミュニケーションコストを抑制できる
- **コードベース認識能力の進化** - LLMがコードを直接探索·読解し、段階的な実装計画を立てる
- **効果的な知識管理** - 優れたプロジェクト管理、ドキュメンテーション能力（ ただし、現段階では人間によるオーケストレーションが必要 ）

Vibe Codingの現状と影響

- **急速な普及** – Fortune 500企業の87%が少なくとも1つ以上のVibe Codingプラットフォームを使用中
- **成果指標** – ガートナーの報告によると、開発者の生産性が37%向上し、新機能のリリース時間が29%短縮され、コード品質は42%向上し、開発者の満足度は53%上昇
- **ツールとプロセスの急速な発展** - モデル、ツール、プロセスのすべてが驚異的な速さで進化している

ロボコが提案するバイブコーディングプロセス

`PRD` > `作業分割` > `TDD実装` > `デプロイ`

PRD作成

> 「AIが皆さんに何をしてくれるかを問うのではなく、皆さんがAIのために何ができるかを問いかけてください」

作業分割

- PRDを基に実装すべき作業を分割
- 作業はできるだけ小さく分け、独立して実装·検証可能であること
- 作業には実装だけでなく、ドキュメンテーション、自動テスト、レビュー、IaC、CI/CD、Ops高度化作業も含む
- Github IssuesやJIRAなどの作業管理もCLIツールを使って自動化可能

TDD実装

- PRDに基づいたテストケースの作成とレビュー
- 小さく実装し、頻繁にテストする
- モジュールや機能、サービス完成時点で統合テストおよびE2Eテストを作成

デプロイ

- CI/CDパイプラインを利用した自動デプロイ
- デプロイ後のモニタリングおよび運用自動化
- デプロイ後のフィードバックをPRDに反映

Vibe Codingとソフトウェアエンジニアリングのレベルアップ

- ドキュメンテーション、テスト自動化、レビュー、IaC、CI/CD、Opsの高度化がサポートされなければ、Vibe Codingは大量のゴミコードを迅速に生成してしまう
- Vibe Codingツールは、ドキュメンテーション、テスト自動化、レビュー、IaC、CI/CD、Opsの高度化に優れた能力を発揮する

> Vibe Codingとソフトウェアエンジニアリングのレベルアップは相互補完的である！

Vibe Codingで達成できるエンジニアリングレベルアップ

Vibe Codingで可能なタスク

> 誰もが必要性を認めながら、なかなか実行されないこと

1. ドキュメンテーション
2. テスト自動化
3. IaCコード生成
4. CI/CDパイプライン
5. Ops自動化
6. セキュリティ自動化

ドキュメンテーション

Vibe Codingで開発する場合

- `PRD` > `タスク分割` > `TDD実装` > `デプロイ` のプロセスを迅速に繰り返す
- この過程で、PRDは常に最新の設計を反映する
- タスク完了時点で、ドキュメントの最新化が常に行われる必要がある

レガシーコードの場合

- ドキュメンテーション作業もプロジェクトとして管理
- README.mdを含め、既存のドキュメントを容易に最新化
- 現在のソースコードだけでなく、コミット履歴、課題トラッカー、Wikiもコンテキストソースとして活用可能
- AIだけでなく、新人や外部協力者のオンボーディング加速にも貢献可能
- 生成されたドキュメントは注意深くレビューする必要がある
    - コードや既存のドキュメントに誤りがある可能性がある

テスト自動化

テスト自動化の実装手順

- テスト自動化も一つの新規プロジェクトとして扱う
- ドキュメントの最新化（要件明確化）から始める
- 静的コードチェッカー > UT > IT > E2Eの順に進める

AIを活用したコード可読性の向上 - 静的コードチェック結果の自動修正

- 静的コードチェッカーのルールは、コードの可読性と密接な関連がある
- AIを活用して、静的コードチェッカーの指摘事項を自動的に修正

Gitフックを使用した迅速かつ頻繁なテスト

- Gitフック機能を利用して、gitイベントで自動的に検証が行われるように設定する
  - pre-commit: Lint, Unit Test
  - pre-push: Integration Test, E2E Test
- huskyなどを使用すると、gitフックを便利に管理可能

IaCコード生成

- 既存インフラ → **IaC(Infrastructure as Code)** 変換
  - 手動で作られたリソースをCLI(cloudcontrol list-resources)で調査
  - 実行コードと同様に、PRD > タスク分割 > 実装 > 検証の段階で進める
  - Terraform / AWS CDKコードを生成
- メリット:
  - インフラ変更履歴の管理
  - 再現可能な環境の提供
  - マルチリージョン/クラウドへの移植性確保

CI/CDパイプライン

- Vibe Codingで **CI/CDワークフローを自動生成**
  - GitHub Actions, GitLab CI, AWS CodePipelineなどを活用して実装
  - 実行コードと同様に、PRD > タスク分割 > 実装 > 検証の段階で進める
- 機能:
  - ビルド/デプロイパイプラインの自動化
  - セキュリティスキャン & 品質検査を含む
- 効果:
  - リリースサイクルの短縮
  - デプロイ失敗率の減少

Ops自動化

- **運用メトリクスベースの自動化:**
  - アラームルール/対応シナリオの自動生成
  - CloudWatch, Prometheus, ELK連携
- 例:
  - CPU超過時に自動スケールアウト
  - 障害発生時にSlack/Teamsへ通知 & 課題生成

セキュリティ自動化

- **AIベースのセキュリティ自動化:**
  - AIを活用して定期的にセキュリティ規定遵守の可否を含む各種セキュリティレビューを実行できる
  - 規定遵守の可否をリアルタイムで監視可能
  - よく知られたコンプライアンスフレームワーク（ OWASP top 10, ISO27001 ）を活用
- 期待効果:
  - 実質的なセキュリティ強化
  - ドキュメンテーションのような面倒な作業のほとんどをAIに任せることができる

運用自動化スクリプト

- 頻繁に繰り返される運用作業のスクリプト自動化:
  - ログ分析 / バックアップ / バッチ処理
  - セキュリティパッチ適用スクリプト
- 期待効果:
  - 運用リソースの最適化
  - ヒューマンエラーの減少

ソフトウェアエンジニアリングレベルアップのためのヒント

- ドキュメンテーションが一番大事
  - ドキュメントはAIと人間の `Single Source Of Truth`
- すべてのコンテキストをAIが容易にアクセス可能なマークダウン形式で作成し、ソースリポジトリと共に、または別途のドキュメントリポジトリを作成して管理
- 共通して参照される内容は、ドキュメントリポジトリを作成し、サブモジュールとして含める
- すべての作業は、基本的なVibe Codingプロセスを同様に遵守すること
  - `PRD作成` > `タスク分割` > `テストを含む実装` > `レビュー/デプロイ`

結論: バイブコーディングとソフトウェアエンジニアリング

- AIを信じるな
- 人を信じるな
- 信頼できるプロセスを作り上げろ！

今後の課題 - レビューをどのように自動化するか？

- Vibe Codingに慣れてくると、最終的に人間による手動レビューがボトルネックとして浮上する
- レビュー/検証をどこまで/どの程度のレベルで自動化するかが、Vibe Codingのレベルを分ける重要な尺度として登場する
- レビュー自動化もAIによる自動化と同様である。原則を立て、それを守れるようにプロンプト戦略を立てる。

ご清聴ありがとうございました