バイトル

ホールインワン開発の夢と現実

～AIコーディングの生産性最大化への道～

2025年10月10日

dip
Labor force solution company

自己紹介

バイトル

黒田

- **所属**: ディップ株式会社
- **SNS**: @judowine (X)
- **専門**: Androidエンジニア
- **好きなもの**: きのこ、作曲、数学

dip
Labor force solution company

バイトル

目次

dip
Labor force solution company

00

はじめに

AIに仕事を任せて楽になりましたか？

バイトル

それとも、逆に振り回されていませんか？

AIコーディングはゴルフに似ている

バイトル

なぜゴルフなのか？

- **試行錯誤が必要**: 一発でホールインワンは狙えない
- **戦略が重要**: 状況に応じて適切なクラブ（ツール）を選ぶ
- **対話的なプロセス**: 一打ごとに状況を見て次の一手を考える
- **環境の理解**: コース（コンテクスト）を読む力が求められる

ゴルフ用語で理解するAI開発

バイトル

基本用語の対応表

ゴルフ用語	AI開発での意味
プレイヤー	私たち開発者
クラブ	AIモデルやツール
ボール	プロンプト
コース	開発のコンテクスト
ホール	目指すゴール

01

夢の「全自動ゴルフ」と最初の池ポチャ

最初の理想：全自動ゴルフロボット

バイトル

PBIを渡せばPRが完成する夢

- PBI（Product Backlog Item）をAIに渡すだけ
- AIが完璧なコードを自動生成
- レビュー不要でそのままマージ
- 開発者は戦略立案に専念できる理想の世界

dip
Labor force solution company

厳しい現実：池ポチャの連発

バイトル

曖昧な指示がもたらした混乱

- OB（Out of Bounds）：要件から大きく逸脱したコード
- 池ポチャ：動かないコード、テストが通らない
- バンカー：なんとか動くが品質が低い
- グリーンに乗らないコードの山が積み上がる

dip
Labor force solution company

教訓：ホールインワンは狙えない

バイトル

AIへの過度な期待が失敗を招く

- 曖昧で自由度の高すぎる指示は機能しない
- 一発で完璧なコードを生成させようとする戦略の失敗
- AIには「段階的なアプローチ」が必要
- まずはグリーンに乗せることから始めるべきだった

dip
Labor force solution company

02

迷走のラウンド ～スコア改善への試行錯誤～

試み1：名選手の「フォーム」を真似させる

バイトル

人間のワークフローをAIに遵守させる

- プロの開発者の手順を詳細に文書化
- AIに厳密なワークフローを指示
- 暴走を防ぐための「型」を提供

結果は…？

dip
Labor force solution company

失敗：形だけでは意味がない

バイトル

思考が伴わないアウトプット

- ワークフローは守るが、本質を理解していない
- 手順通りでも的外れなコードが生成される
- アウトプットのブレは改善されず
- 「なぜその手順なのか」をAIは理解できていなかった

dip
Labor force solution company

試み2：「スパルタコーチ」になる

バイトル

AIの思考プロセスを徹底的にレビュー

- AIの出力を一つ一つ細かくチェック
- マイクロマネジメントで方向修正
- 人間が常に監視・介入する体制

結果は…？

本末転倒：ボトルネックは人間だった

バイトル

精度は上がったが、負荷が爆増

- コードの品質は確かに向上した
- しかし、人間のレビュー負荷が爆発的に増加
- 「自分で書いた方が早い」状態に
- AIで効率化するはずが、逆に生産性が低下

dip
Labor force solution company

03

ブレークスルー！「ヤーデージブック」と監督の役割

発想の転換

バイトル

AIは「指示待ちマシーン」ではない

- これまで: AIに細かく指示を出す存在
- これから: AIを自律的に判断できるパートナーに
- 必要なのは「管理」ではなく「環境整備」
- AIが自分で考え、判断できる土台を作る

dip
Labor force solution company

ヤーデージブック作戦

ゴルフのヤーデージブックとは？

プロゴルファーが使う、コースの詳細情報が記載されたガイドブック

AI開発への応用

バイトル

ヤーデージブック作戦の4つの柱

1. **コースのシンプル化** - アーキテクチャの統一
2. **ローカルルールの制定** - 制約の明文化
3. **参照ページの限定** - コンテキストの提供
4. **指示の構造化** - Gherkin記法など

dip
Labor force solution company

役割の変化：コーチから監督へ

バイトル

Before: スパルタコーチ

- 一つ一つの動きを細かく指示
- AIの出力を常に監視
- 汗だくで現場に張り付く

After: 戦略的な監督

- 大局的な戦略を立案
- 環境を整備し、方針を示す
- AIが自律的に動ける仕組みを作る

dip
Labor force solution company

具体例：Claude Codeでの監督の役割

バイトル

従来（スパルタコーチ）

- 「この関数をこう書いて」と細かく指示
- AIの出力を一行ずつレビュー
- 修正指示の繰り返し

監督としてのアプローチ

- ヤーデージブック（アーキテクチャ、制約）を整備
- 「このユーザーストーリーを実装して」と意図を伝える
- AIが自律的に判断・実装
- 人間は結果を検証し、方針を調整

dip
Labor force solution company

指示の構造化：自然言語は最適なDSLではない

バイトル

自然言語の限界

- 曖昧さが残る
- コンテキストの共有が難しい
- 意図が正確に伝わらない

dip
Labor force solution company

コードベースでの対話が最適

バイトル

AIとのコミュニケーション戦略

- **実際の振る舞いを見せる**: テストコード、サンプル実装
- **構造化された指示**: Gherkin記法（Given-When-Then）
- **コードが語る**: アーキテクチャ、パターン、制約がコードに表れる
- AIは実際のコードから学習し、一貫性のある実装を生成

04

見えてきた最大の壁 〜AIではなく「開発」の本質〜

それでも残る疑問

バイトル

なぜまだ究極の理想には届かないのか？

- ヤーデージブック作戦で生産性は向上した
- しかし、PBI→PRの完全自動化は実現していない
- 依然として人間の介入が必要な場面がある
- 何が本当の壁なのか？

dip
Labor force solution company

本質的な気づき

バイトル

本当の壁は、AIではなく「人間」だった

- 人間自身も、実際にやってみるまで要求を完全には理解していない
- 「作ってみて初めてわかる」ことが多い
- これは開発の不変の真理
- AIの問題ではなく、ソフトウェア開発の本質的な課題

dip
Labor force solution company

実装＝発見のプロセス

バイトル

設計と実装は要求を発見するための「対話」

- 要件定義は「完璧な設計図」ではない
- 実装する中で隠れた課題が見えてくる
- エッジケース、パフォーマンス、UX...
- コースを歩いて初めて気づくハザード（罠）がある

dip
Labor force solution company

AI時代に求められる設計能力

不確実性をナビゲートする力

- 完璧な設計図を描く能力ではない
- AIと共に不確実な状況を探索していく能力
- 実装しながら要求を明確化していくプロセス
- これこそが、AI時代の真の「設計力」

05

AI-DLC：実践への道 〜AIの生産性を増す=人間の生産性を増す？〜

SDLC（Software Development Lifecycle）とは

バイトル

ソフトウェア開発のライフサイクル

- 要件定義 → 設計 → 実装 → テスト → 運用保守
- アジャイル、スクラム、ウォーターフォールなど様々な手法
- 従来は人間が主体となって各フェーズを進める

従来の手法の課題

- 数週間〜数ヶ月単位の長いイテレーション
- 手動のワークフローと厳格な役割分担
- AIの速度や柔軟性と合致しない

dip
Labor force solution company

AI-DLC（AI-Driven Development Lifecycle）

バイトル

AIを中心に据えた開発ライフサイクル

- 既存の手法に「後付け」ではなく、第一原理から再設計
- 数時間〜数日単位の高速イテレーション
- AIが主導し、人間が戦略的に監督する

重要な原則

- **会話の方向を逆転**: AIが対話を開始・推進
- **設計技法をコアに統合**: DDD、BDD、TDDを組み込む
- **人間とAIの共生**: 検証・意思決定は人間が保持

dip
Labor force solution company

AI-DLCの3つのフェーズ

バイトル

1. Inception Phase（構想フェーズ）

Intent（意図）を捉え、Unit（作業単位）に分解

2. Construction Phase（構築フェーズ）

ドメイン設計から論理設計、コード生成とテスト

dip
Labor force solution company

AI-DLCの3つのフェーズ（続き）

バイトル

3. Operations Phase（運用フェーズ）

デプロイメント、監視、インシデント管理

Inception Phase：不確実性をナビゲートする

バイトル

第4章で述べた「設計能力」の実践

- 完璧なPBIを最初から作ることはできない
- AIとの対話を通じて要求を「発見」していく
- これが「実装＝発見」プロセスの第一歩
- 曖昧な意図（Intent）から始める
- AIが質問し、人間が答える対話
- 隠れた要件、リスク、制約を明らかにする

dip
Labor force solution company

AI-DLCを実践してみた

バイトル

AWSの公式リソース

AI-DLCについて詳しく知りたい方はこちら：

https://aws.amazon.com/jp/blogs/news/ai-driven-development-life-cycle/

dip
Labor force solution company

実践：簡単なサンプルアプリで試してみた

バイトル

アプローチ

- Inception Phaseを使ってユーザーストーリーを作成し、そこからPBIを作成
- AIとの対話を通じて要件を明確化
- 実際の体験から見えてきたこと

dip
Labor force solution company

サンプルアプリ：connpass出会い管理アプリ

バイトル

Intent（意図）

connpass APIを使用して「参加したconnpassのイベントで誰に会ったかを記録・管理し、次回会う時に備える」

解決したい課題

- イベントに参加しても、後で「あの人誰だっけ？」となる
- 「何を話したっけ？」と会話の内容を忘れてしまう
- 次回会う時に備えて、情報を整理しておきたい

dip
Labor force solution company

Inception Phaseで生成された成果物

バイトル

AIとの対話で得られたもの

- **ユーザーストーリー**: 6件
- **エピック**: 5件
- **ユニット**: 11件
- **ユニットの依存関係**: 自動生成
- **PBI（Product Backlog Item）**: ユニットから自動生成

従来との違い

手動で作成すると数時間〜数日かかる作業が、対話を通じて数十分で完成

dip
Labor force solution company

実践から見えてきたこと

バイトル

従来のAI後付けSDLC

- **レビュー地獄**: 実装前にAIが大量の設計書を生成
- 人間が全ての設計書をレビュー
- スパルタコーチ状態に逆戻り

dip
Labor force solution company

AI-DLCのアプローチ

バイトル

対話による必要最小限の情報抽出

- AIが「わかること・わからないこと」を自己申告
- 対話を通じて必要な情報だけを引き出す
- 構造化された成果物（ユーザーストーリー、ユニット、依存関係）
- 人間は重要な判断ポイントだけを検証する「監督」へ

dip
Labor force solution company

まとめ：ホールインワン開発への道

バイトル

学んだこと

1. **AIに完璧を求めない**: 段階的なアプローチが鍵
2. **形だけ真似ても意味がない**: 思考プロセスの理解が重要
3. **監督としての役割**: マイクロマネジメントから戦略的監督へ
4. **不確実性のナビゲート**: AIとの対話で要求を発見する

dip
Labor force solution company

AI時代の開発者に求められるもの

バイトル

新しい「設計能力」の定義

- 完璧な設計図を描く能力ではない
- AIと共に不確実性を探索し、ナビゲートする力
- これが真の「設計能力」

dip
Labor force solution company

バイトル

ありがとうございました

dip
Labor force solution company