

# 第1章から第7章 マインドマップ

---

## 全体構造

mindmap

root((開発生産性の誤解と正しさ))

第1部: 突然求められる開発生産性の重圧

第1章: 明日までに改善して！と言われる重圧

1-1: エンジニアが感じる「開発生産性」という言葉の重圧

突然の重圧

経営陣からの数字要求

エンジニアの困惑

1-2: スピード優先がもたらす負のスパイラル

PMからの緊急案件

バグの連鎖

チームの疲弊

1-3: 「このままで無理だ」という気づき

持続不可能な状況

根本的な問題の認識

手法1: 開発生産性の危険な落とし穴チェックリスト

第2章: 開発生産性の目的を求められる重圧

2-1: 単なるスピードや数値を超えた生産性の多角的理解

深夜の自問自答

様々な指標の検討

コード行数・機能リリース数の限界

2-2: エンジニアとして他者の視点を想像する重要性

PM・QA・サポートの視点

多角的な理解

2-3: 持続可能な開発とは何かを模索する

長期的な視点

価値創出の重要性

手法2: ステークホルダー視点マッピング

第3章: 測れない生産性を求められる重圧

3-1: 技術的負債の蓄積と長期的な開発速度の関係

バグ対応の忙しさ

技術的負債の現実

開発速度の低下

3-2: 「今は動くから」の先にある持続不可能な開発環境

短期的な判断の危険性

長期的な影響

3-3: エンジニアとしての説明責任を果たす方法

可視化の重要性

ビジネス側への説明

手法3: 測れない価値の可視化ダッシュボード

第2部: チームでの問題解決 - 実践と改善

第4章: 重圧から生み出される小手先の指標

4-1: 小手先の指標に依存せずに問題を特定する手法

ワークショップの準備

問題の本質を見極める

4-2: それぞれの立場から見た「痛み」を共有し理解する対話

本音の爆発

痛みのマッピング

チーム全体での対話

4-3: チーム全体で合意できる本質的な価値の発見

共通の価値基準

合意形成

手法4: 脱・形骸化指標リスト

第5章: 開発リソースは足りないのに増え続ける重圧

5-1: AIを「コードを早く書く道具」以上のパートナーに

新たな課題の顕在化

AIツールへの疑問

AIの戦略的活用

5-2: AIによる技術的負債検出と品質向上の実践

技術的負債の検出

品質向上への活用

5-3: 運用業務も効率化する

運用業務の自動化

効率化の実現

5-4: 新規開発とリファクタリングの両立を支えるAI活用法

両立の重要性

AIによる支援

5-5: 心理的安全性が支える開発生産性

チームの心理的安全性

生産性への影響

手法5: AIとの効果的協働ガイド

手法6: プロンプトエンジニアリング事例集

手法7: 心理的安全性評価・向上ガイド

手法8: 持続可能なペース配分計画

第3部: 開発生産性を事業貢献につなげる

第6章: エンジニアだけが生産性を求められる重圧

6-1: エンジニア視点とビジネス視点の融合

経営層からの評価依頼

他部署との対話

視点の融合

6-2: PMやPdMと共に創る「共通の価値基準」

共通価値の創出

協働の重要性

6-3: 工数・工期・事業KPIを包括した生産性の新しい評価軸

包括的な評価

多角的な指標

手法9: 多角的生産性評価マトリクス

手法10: ステークホルダー対話ガイド

第7章: 開発生産性をあげるだけのエンジニアからの脱却

7-1: 技術的負債返済と新規開発の両立による長期的成功

1年後の驚異的な成果

組織全体での価値認知

7-2: AIとの戦略的協働がもたらした開発速度5倍の実現

AI活用の成果

開発速度の向上

7-3: 数値を超えた「価値創出」を評価する組織文化の確立

価値創出の評価

組織文化の変化

7-4: 湊の成長: 技術者からビジネス視点も持つエンジニアリーダーへ

個人の成長

リーダーシップの確立

手法11: 持続可能な開発成功事例集

手法12: AI協働実践ガイド

# 各章の詳細マインドマップ

第1章: 明日までに改善して！と言われる重圧

mindmap

root((第1章))

1-1: エンジニアが感じる「開発生産性」という言葉の重圧

突然の重圧

田中部長からの呼び出し

経営陣からの数字要求

エンジニアの困惑

開発生産性の定義の曖昧さ

コード行数?

リリース頻度?

ストーリーポイント?

チームの反応

山田の困惑

佐藤の不安

チーム全体の疲労

1-2: スピード優先がもたらす負のスパイラル

PMからの緊急案件

飛鳥PMの要求

来月末リリース必須

仕様変更の連続

バグの連鎖

リリース後のバグ

QA部の報告

モグラ叩き状態

チームの疲弊

残業の増加

モチベーション低下

持続不可能な状況

1-3: 「このままで無理だ」という気づき

根本的な問題の認識

技術的負債の蓄積

品質の低下

チームの疲弊

改善への決意

一人で抱え込まない

チーム全体での取り組み

手法1: 開発生産性の危険な落とし穴チェックリスト

## 第2章: 開發生産性の目的を求められる重圧

mindmap

root((第2章))

2-1: 単なるスピードや数値を超えた生産性の多角的理解

深夜の自問自答

開發生産性って何?

様々な指標の検討

指標の限界

コード行数: 無駄なコードで上がる

機能リリース数: 品質を犠牲にすれば上がる

ストーリーポイント: 見積もり精度に依存

DORAメトリクス

デプロイ頻度

変更のリードタイム

変更失敗率

回復時間

物的生産性 vs 付加価値生産性

物的生産性: 数値化しやすい

付加価値生産性: 理想だが測りにくい

2-2: エンジニアとして他者の視点を想像する重要性

PMの視点

スケジュール管理

顧客要求への対応

QAの視点

品質保証

バグ検出

サポートの視点

顧客対応

問題解決

2-3: 持続可能な開発とは何かを模索する

長期的な視点

短期的な成果 vs 長期的な価値

技術的負債の返済

価値創出の重要性

ビジネス価値

ユーザー価値

手法2: ステークホルダー視点マッピング

## 第3章: 測れない生産性を求められる重圧

mindmap

root((第3章))

3-1: 技術的負債の蓄積と長期的な開発速度の関係

バグ対応の忙しさ

リリース後の混乱

緊急対応の連続

チームの疲弊

技術的負債の現実

開発速度の低下

年間20-40%の速度低下

コードの複雑化

開発速度の可視化

過去のデータ分析

技術的負債の影響

3-2: 「今は動くから」の先にある持続不可能な開発環境

短期的な判断の危険性

今は動くから大丈夫

先送りされる問題

長期的な影響

開発速度の低下

品質の悪化

チームの疲弊

3-3: エンジニアとしての説明責任を果たす方法

可視化の重要性

技術的負債の可視化

開発速度の可視化

影響の可視化

ビジネス側への説明

技術用語の翻訳

ビジネスインパクトの説明

投資対効果の説明

手法3: 測れない価値の可視化ダッシュボード

## 第4章: 重圧から生み出される小手先の指標

mindmap

root((第4章))

4-1: 小手先の指標に依存せずに問題を特定する手法

ワークショップの準備

一人で抱え込まない

チーム全体での対話

本音を引き出す工夫

問題の本質を見極める

表面的な指標の限界

根本原因の分析

4-2: それぞれの立場から見た「痛み」を共有し理解する対話

本音の爆発

山田の痛み

高橋の痛み

飛鳥PMの痛み

佐藤の痛み

痛みのマッピング

付箋を使った整理

痛みの分類

関連性の可視化

チーム全体での対話

心理的安全性の確保

全員の意見を尊重

4-3: チーム全体で合意できる本質的な価値の発見

共通の価値基準

本質的な価値の特定

合意形成

改善への道筋

優先順位の決定

具体的なアクション

手法4: 脱・形骸化指標リスト

## 第5章: 開発リソースは足りないのに増え続ける重圧

mindmap

root((第5章))



### 5-1: AIを「コードを早く書く道具」以上のパートナーに

- 新たな課題の顕在化

- 品質と速度の両立

- リソースの制約

- AIツールへの疑問

- GitHub Copilot

- 単なるコード生成？

- AIの戦略的活用

- コード生成だけでない活用

- 品質向上への活用

- 技術的負債の検出

### 5-2: AIによる技術的負債検出と品質向上の実践

- 技術的負債の検出

- AIによる自動検出

- コードレビューの支援

- 品質向上への活用

- バグ検出の支援

- テスト生成の支援

### 5-3: 運用業務も効率化する

- 運用業務の自動化

- ログ分析

- 監視業務

- 効率化の実現

- 時間の確保

- 開発への集中

### 5-4: 新規開発とリファクタリングの両立を支えるAI活用法

- 両立の重要性

- 新規開発の継続

- リファクタリングの実施

- AIによる支援

- 両立の実現

- 効率的な開発

### 5-5: 心理的安全性が支える開発生産性

- チームの心理的安全性

- 意見を言いやすい環境

- 失敗を許容する文化

- 生産性への影響

- イノベーションの促進

- チームの成長

### 手法5: AIとの効果的協働ガイド

手法6: プロンプトエンジニアリング事例集

手法7: 心理的安全性評価・向上ガイド

手法8: 持続可能なペース配分計画

## 第6章: エンジニアだけが生産性を求められる重圧

mindmap

root((第6章))

6-1: エンジニア視点とビジネス視点の融合

経営層からの評価依頼

経営会議での発表

成果の説明

他部署との対話

営業部: 顧客満足度の向上

サポート部: バグ報告の減少

経営層: ビジネスインパクト

視点の融合

エンジニア視点

ビジネス視点

統合的な視点

6-2: PMやPdMと共に創る「共通の価値基準」

共通価値の創出

エンジニアとPMの協働

価値基準の合意

協働の重要性

相互理解

共通目標

6-3: 工数・工期・事業KPIを包括した生産性の新しい評価軸

包括的な評価

工数の評価

工期の評価

事業KPIの評価

多角的な指標

技術指標

ビジネス指標

統合指標

手法9: 多角的生産性評価マトリクス

手法10: ステークホルダー対話ガイド

## 第7章: 開発生産性をあげるだけのエンジニアからの脱却

mindmap

root((第7章))

7-1: 技術的負債返済と新規開発の両立による長期的成功

1年後の驚異的な成果

開発速度300%向上

バグ発生率70%減少

技術的負債50%削減

顧客満足度向上

組織全体での価値認知

CEOの評価

全社標準への展開

予算化の実現

7-2: AIとの戦略的協働がもたらした開発速度5倍の実現

AI活用の成果

開発速度の向上

品質の向上

効率化の実現

開発速度5倍の実現

技術的負債返済との両立

持続可能な開発

7-3: 数値を超えた「価値創出」を評価する組織文化の確立

価値創出の評価

数値だけではない評価

長期的な価値

組織文化の変化

評価基準の変更

文化の変革

7-4: 湊の成長: 技術者からビジネス視点も持つエンジニアリーダーへ

個人の成長

技術スキル

ビジネススキル

リーダーシップ

リーダーシップの確立

チームの牽引

組織への影響

手法11: 持続可能な開発成功事例集

手法12: AI協働実践ガイド

# 主要テーマの関連性

```
graph TD
    A["A[開発生産性の誤解]"] --> B["B[第1部: 気づきと混乱]"]
    A --> C["C[第2部: 実践と改善]"]
    A --> D["D[第3部: 事業貢献]"]

    B --> B1["B1[第1章: 重圧の認識]"]
    B --> B2["B2[第2章: 多角的理解]"]
    B --> B3["B3[第3章: 可視化]"]

    C --> C1["C1[第4章: 痛みの共有]"]
    C --> C2["C2[第5章: AI活用]"]

    D --> D1["D1[第6章: 視点の融合]"]
    D --> D2["D2[第7章: 価値創出]"]

    B1 --> C1
    B2 --> C1
    B3 --> C2
    C1 --> D1
    C2 --> D2
    D1 --> D2

    style A fill:#f9f,stroke:#333,stroke-width:4px
    style B fill:#bbf,stroke:#333,stroke-width:2px
    style C fill:#bfb,stroke:#333,stroke-width:2px
    style D fill:#fbf,stroke:#333,stroke-width:2px
```