● 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

年次レビューへの準備

AI活用を始めてから1年が経過した。湊のチームは驚異的な成果を上げていた。

年次レビューの数値発表で、湊は緊張しながらも自信を持って報告した。

1年後の成果

「開発速度が300%向上しました。バグ発生率は70%減少しています。技術的負債は50%削減され、顧客満足度も業界平均を大きく上回っています」

会議室にいた全員が驚いた表情を見せた。開発速度300%向上。それは単なる数値の改善ではなく、チーム全体の取り組みが実を結んだ証だった。

「どうやって実現したんですか？」

他チームのリーダーが質問した。

湊は画面にグラフを映しながら説明を始めた。

「1年前、同じ規模の機能開発に平均して7日かかっていました。でも、技術的負債を継続的に返済し、AIを戦略的に活用することで、今は2.3日で開発できるようになりました。これが開発速度300%向上の根拠です」

湊は次のグラフに切り替えた。

「バグ発生率について。1年前は月に15件のバグがリリース後に発見されていました。でも、第4章で導入した自動テストと、第5章で導入したAIによるテストケース生成により、開発段階でバグを発見できるようになりました。結果として、リリース後のバグは月に4.5件に減少しました。これが70%減少の根拠です」

湊は最後のグラフを見せた。

「技術的負債について。1年前、コードの循環的複雑度は平均15でした。でも、週1回のリファクタリング時間を確保し、AIによる技術的負債検出を活用することで、平均7.5まで削減できました。これが50%削減の根拠です」

「特別なことはしてません。基本を丁寧に。技術的負債の返済と新規開発の両立を続けてきただけです」

湊は淡々と答えた。でも内心は達成感でいっぱいだった。1年前、第1章で失敗した時、一人で抱え込んでいた自分。第2章で各職種の視点を学び、第3章でデータ分析を始め、第4章でチーム全体で問題を共有し、第5章でAIを戦略的に活用し、第6章でビジネス視点を獲得した。そして今、第7章で成果を実現できた。それは、一夜にして変わったわけではない。少しずつ、対話を通じて積み重ねてきたものだった。

山田も湊を評価していた。

「湊さん、君が種を蒔いた木が、大きく育ったね。技術的負債の返済に時間をかけることの価値を、数値で証明できた」

佐藤らチームメンバーも成長していた。技術的なスキルだけでなく、ビジネス視点も持つようになっていた。

「技術的負債の返済により、開発速度が年間20-40%向上。これは同じ機能を開発するのに、以前より少ない工数で済むことを意味します」

湊はデータを見せながら続けた。

「バグ発生率が70%減少したことで、サポート対応の工数も大幅に削減できました。顧客満足度も向上し、営業部からの評価も高まっています」

組織全体への影響

全社会議での事例発表。

CEOからの評価

CEOが湊のチームの取り組みを評価した。

「湊チームの取り組みを全社標準にしたい。技術的負債返済が『必要な投資』として予算化されるべきだ」

CEOの言葉に、湊は深く頷いた。1年前、「生産性って何？」と悩んでいた自分が、今は組織全体に影響を与える存在になっていた。

「リファクタリング時間が正式に確保される制度も作ります。技術的負債の返済は、短期的なコストではなく、長期的な投資です」

田中部長が続けた。

「湊君、君が変化の起点になったね。この取り組みが、組織全体に良い影響を与えている」

「一人では無理でした。チーム全体の力です」

湊は正直に答えた。

技術的負債返済が「必要な投資」として予算化された。リファクタリング時間が正式に確保される制度が確立された。

「測れない価値」が「測れる価値」になった。

湊は実感していた。技術的な改善が、ビジネス価値につながる。それを数値で証明できたことが、組織全体の変化につながった。

湊は改めて考えてみた。開発生産性の本質は「開発組織が『いつまでに何ができるか』と語った約束が、どれだけ信頼できるか」である。技術的負債の返済は、単なる「負債の返済」ではなく、「将来の見積精度向上とリードタイム分散低減への投資」として位置づける必要がある。これにより、内部品質投資が事業価値に直接つながることが明確になった。開発生産性の本質が「信頼性（Predictability / Reliability）」であることが、組織全体で理解されるようになった。

解説：技術的負債返済と新規開発の両立による長期的成功

湊のチームが実現したように、技術的負債返済と新規開発の両立は可能です。

なぜ技術的負債返済と新規開発の両立が困難なのか

技術的負債の返済に時間をかけると、新規開発の時間が減ります。短期的な成果を求められると、技術的負債の返済が後回しになります。技術的負債の影響は数ヶ月〜数年後に顕在化するため、その価値を理解してもらうのが困難です。

技術的負債返済と新規開発の両立方法

時間配分の最適化：

- 新規開発と技術的負債返済の時間配分を最適化する
- 週1回のリファクタリング時間を確保する
- 技術的負債のリスクスコアに基づいて優先順位を決定する

継続的な返済：

- 一度に全てを返済するのではなく、継続的に返済する

- ボーイスカウトルールを実践する（コードを触ったら、少しでも良くしてから離れる）
- 20%ルールを活用する（開発時間の20%を技術的負債返済に充てる）

価値の可視化：

- 技術的負債の影響をコスト換算で示す
- 開発速度の向上を数値で証明する
- バグ発生率の減少を具体的に示す

長期的成功の実現

技術的負債の削減により開発速度が向上します。湊のチームでは、技術的負債50%削減により開発速度300%向上を実現しました。バグ発生率の減少により、サポート対応の工数も大幅に削減できます。組織文化の変化により、技術的負債返済が「必要な投資」として認識されます。

詳細な手法については、章末の「手法11：持続可能な開発成功事例集」を参照してください。

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との協働の進化

技術的負債検出の自動化システムが構築完了した。毎週自動でコードベース全体をスキャンし、リスクスコアに基づく優先順位付けが可能になった。

「AIが第6のメンバーみたいになってる」

山田が笑いながら言った。

「確かに。AIとの協働が日常になって、開発の効率が格段に上がった」

湊も同意した。

AIペアプログラミングが日常化していた。AIがコードを提案し、人間がレビューして修正する。そのプロセスで、開発速度が5倍に向上していた。

「人間とAIの完璧な役割分担ができてるね」

山田が続けた。

「AIが単純な作業を担い、人間は創造的な仕事に集中できる。これが理想的な使い方だと思う」

5名のチーム全員がAIとの協働スキルを習得していた。技術的負債検出、テストケース生成、コードレビュー支援。AIを多角的に活用することで、開発全体の効率化が実現していた。

AIがチームの一員に

AIとの協働が日常化し、チームの一員として定着していた。

他チームとの比較検証

半年前、速度重視で開始された別チームがあった。最初の3ヶ月は湊チームより速いペースで開発を進めていた。

「あのチーム、すごく速いね」

佐藤が心配そうに言った。

「でも、このまま続けられるかな」

湊は深く考え込んだ。確かに、速度重視のチームは最初は速い。でも、技術的負債を無視して開発を進めると、後で大きな問題が発生する。第1章で自分が経験した失敗を思い出した。急いでリリースした結果、バグだらけのシステムを世に出してしまった。チームは徹夜で障害対応に追われ、ユーザーからのクレームも殺到した。

3ヶ月後、予想通り別チームは失速していた。バグ対応、仕様変更で混乱。技術的負債の蓄積により開発速度が50%以下に低下していた。

湊はデータを比較した。速度重視のチームは、最初の3ヶ月は1週間で機能を開発できていた。でも、4ヶ月目から開発速度が低下し始め、6ヶ月後には同じ機能の開発に2週間かかるようになっていた。一方、湊のチームは、最初は1週間かかっていたが、技術的負債を継続的に返済することで、徐々に開発速度が向上し、1年後には2.3日で開発できるようになっていた。

「やっぱり、持続可能な開発が重要だったんだね」

山田がデータを見せながら言った。

「速度重視のチームは最初は速いけど、技術的負債が蓄積すると失速する。コードの複雑度が増し、どこを触っても影響範囲が広がる。バグ修正のたびに別の場所が壊れる。モグラ叩きのような状態になってしまう。でも、私たちのチームは技術的負債を継続的に返済してきたから、安定して高速な開発ができてる」

データが証明していた。持続可能な開発の優位性が明確に示されていた。

データが示す真実

湊は第3章で学んだ「技術的負債の蓄積で開発速度が年間20-40%低下する」という事実を思い出した。でも、それを逆に活用すれば、技術的負債を返済することで開発速度を向上させられる。それが、1年間の取り組みで証明できた。

経営層も結果を評価していた。

「やはり湊方式が正しかった。短期的な速度よりも、持続可能性が大事だということだね」

解説：AIとの戦略的協働がもたらした開発速度5倍の実現

湊のチームが実現したように、AIとの戦略的協働により開発速度を大幅に向上させることができます。

AI活用の段階的進化

AI活用は段階的に進化します。最初はコード生成から始まり、次に技術的負債検出、テスト自動化へと発展します。人間とAIの役割分担を明確にすることで、より効果的な協働が可能になります。

AI協働による開発速度向上のメカニズム

技術的負債検出の自動化：

- 毎週自動でコードベース全体をスキャン
- リスクスコアに基づく優先順位付け
- 継続的な監視により、技術的負債の蓄積を防ぐ

AIペアプログラミング：

- AIがコードを提案し、人間がレビューして修正する
- 開発速度が5倍に向上
- 人間とAIの完璧な役割分担により、品質も保てる

人間とAIの役割分担：

- AIが得意なこと：データ分析、パターン検出、網羅的チェック
- 人間が得意なこと：コンテキスト理解、創意工夫、意思決定

持続可能な開発の実現

速度重視のチームは最初は速いが、技術的負債の蓄積により失速します。持続可能な開発を実践することで、長期的には安定した開発速度を実現できます。データによる証明により、持続可能な開発の優位性が明確になります。

詳細な手法については、章末の「手法12：AI協働実践ガイド」を参照してください。

7-3 数値を超えた「価値創出」を評価する組織文化の確立

評価制度の定着

新しい評価制度が定着していた。ビジネスインパクト、持続可能性、効率性の3つの軸で評価する制度が、組織全体に広がっていた。

「数値だけで測れない価値も評価されるようになった」

飛鳥PMが満足そうに言った。

「技術的負債の返済が『必要な投資』として認識される。これが重要だったんだね」

山田も同意した。

「リファクタリング時間が正式に確保される制度も確立された。これで、技術的負債の返済が継続的に進められる」

湊は組織文化の変化を実感していた。1年前、「生産性って何？」と悩んでいた自分が、今は組織全体の評価制度を変える存在になっていた。

「測れない価値」が「測れる価値」になった。それは数値だけでなく、組織文化の変化でもあった。

全社標準化への道

CEOからの要請により、湊チームの取り組みが全社標準になることが決まった。

「湊チームの取り組みを、他のチームにも展開したい。ノウハウを体系化して、段階的に広げていこう」

田中部長が説明した。

「まず、ノウハウを体系化します。その後、他チームに展開していきます」

湊は計画を説明した。

他チームからの関心も高かった。

「うちでもやってみたい。どうやって始めればいい？」

「まず、チーム全体で現状を把握することから始めます。その後、段階的に改善を進めていきます」

湊は他チームからの質問に答えながら、組織全体への波及効果を実感していた。

解説：数値を超えた「価値創出」を評価する組織文化の確立

湊のチームが実現したように、数値を超えた価値創出を評価する組織文化を確立することができます。

開発生産性の本質的な定義

開発生産性の正体は、開発組織の速度や件数ではなく、「事業責任者やPdMが『この組織の言葉を信じて意思決定できるか』という信頼性（Predictability / Reliability）である。または、事業責任者やPdMから開発組織への満足度である。」

つまり、開発生産性の本質は「開発組織が『いつまでに何ができるか』と語った約束が、どれだけ信頼できるか」なのです。

内部品質投資の再定義

技術的負債の返済は、単なる「負債の返済」ではなく、「将来の見積精度向上とリードタイム分散低減への投資」として位置づける必要があります。これにより、内部品質投資が事業価値に直接つながることが明確になります。

なぜ数値指標だけでは不十分なのか

数値だけで測れない価値があります。技術的負債の返済、チームの成長、組織への貢献。これらは数値だけで測るのが困難です。組織文化の重要性から、数値を超えた価値を評価する組織文化が必要です。

さらに、開発組織が見ているもの（リードタイム、デプロイ頻度、変更失敗率、MTTR）と、事業責任者・PdMが見ているもの（計画信頼性・予測可能性）は異なります。Four Keysなどの指標は開発組織の内部安定性を測りますが、事業計画の信頼性は直接測りません。この違いを理解し、多角的な評価軸を設けることが重要です。

数値を超えた価値創出の評価方法

多角的な評価軸：

- ビジネスインパクト：売上・顧客満足度への貢献
- 持続可能性：技術的負債管理、チーム成長
- 効率性：開発速度、品質

定性的な評価：

- 数値だけでなく、チームの成長、組織への貢献も評価
- 長期的な視点：短期的な数値だけでなく、長期的な価値も評価

開発生産性ツリーを用いた価値創出の評価：

- 財務（P/L・B/S）と開発行動を一本で接続した評価方法
- 予算（財務）起点で開発投資を分解し、価値創出工数と価値維持工数を区別して評価
- 開発生産性ツリーの構造（予算 → 人件費 → 開発区分 → 工数 → 施策 → バリューストリーム → 開発ライフサイクル）を用いて、投資がどの施策に、どの工程で、どの開発フェーズで消費されているかを可視化
- 価値創出工数（新規開発・エンハンス開発）と価値維持工数（保守開発・運用・管理業務）を区別することで、数値を超えた価値創出を構造的に評価できる
- BigQueryとLookerを用いて実際のデータを可視化・分析し、説明可能な形で価値創出を評価

評価制度の設計：

- 数値だけでなく、定性的な評価も含める
- 定期的に評価軸を見直し、改善する

組織文化の確立

評価制度の定着により、技術的負債返済が「必要な投資」として認識されます。リファクタリング時間の制度化により、リファクタリング時間が正式に確保されます。全社標準化により、湊チームの取り組みが組織全体に広がります。

詳細な手法については、章末の「手法11：持続可能な開発成功事例集」を参照してください。

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

メンター湊の誕生

新卒エンジニアの田村が配属された。田村は意気込んで湊に話しかけた。

「湊さん、コードをバリバリ書いて活躍したいです！」

湊は微笑みながら答えた。

「まずは、なぜそのコードが必要なのか考えよう。技術だけでなく、ビジネスの視点も持つことが大事だよ」

田村は驚いた様子で聞き返した。

「ビジネスの視点...ですか？」

「うん。エンジニアは技術的なスキルだけじゃなく、ビジネス視点も持つ必要がある。なぜその機能が必要なのか、それがどうビジネスに貢献するのか。それを理解することで、より価値のあるコードが書けるようになる」

湊は1年前の自分を思い出していた。「生産性って何？」と悩んでいた自分。でも今は、後輩にアドバイスできる立場になっていた。

「エンジニアって、こんなに幅広く考えるんですね」

田村は感心した様子で言った。

「うん。技術だけでなく、ビジネス視点も持つ。それがエンジニアリーダーとしての成長なんだよ」

湊の成長実感

新卒の田村との対話で、湊は自身の成長を実感していた。

湊は振り返った。1年前、第1章で失敗した時、一人で抱え込んでいた。田中部長から「開発生産性を向上させてくれ」と言われ、どうすればいいのかわからず、ただコードを書く速度を上げようとした。でも、それは間違いだった。

第2章で、各職種の視点を学んだ。山田さんとの対話を通じて技術的負債の重要性を知り、高橋さんとのランチで品質の視点を学び、飛鳥PMとの議論でビジネス視点を理解した。この時、初めて気づいた。生産性という言葉は、立場によって意味が変わる。エンジニアにとっての生産性と、PMにとっての生産性、QAにとっての生産性。それらは対立するものではなく、補完し合うものなのだと。

第3章で、データ分析を始めた。Git履歴を3年分さかのぼり、JIRAチケットの完了時間を集計した。スプレッドシートにグラフを描くと、明確な下降トレンドが見えた。技術的負債の蓄積で開発速度が50%低下している。でも、それをどう伝えれば理解してもらえるのか。一人で作業を続けても、誰も見てくれないかもしれない。でも、山田さんが「一人で抱え込む必要はない」と声をかけてくれた。

第4章のワークショップで、チーム全体で問題を共有した。各職種の痛みを共有し、共通の課題を見つけた。形骸化した指標に依存せず、本質的な価値を創出することが重要だと理解した。それは、一人で抱え込んでいた時には見えなかった道だった。

第5章で、AIを戦略的に活用した。単にコード生成を速くするのではなく、技術的負債検出やテストケース生成にAIを活用することで、開発全体の効率化を実現した。AIは第6のメンバーとして、チームに欠かせない存在になった。

第6章で、ビジネス視点を獲得した。経営会議で、技術的な改善をビジネス価値に変換して説明した。開発速度の向上が、売上や顧客満足度にどうつながるのか。それを数値で示すことで、経営層にも理解してもらえた。

そして今、第7章で成果を実現できた。1年前、「生産性って何？」と悩んでいた自分。今は5名のチームのリーダーとして、組織・事業全体を俯瞰できる視点を獲得していた。

「湊君、次は部門横断のテックリードに昇格を検討したい」

田中部長が提案した。

「責任は重いですが、やりがいもありそうです」

湊は正直に答えた。

山田も湊を評価していた。

「湊さん、君が種を蒔いた木が、大きく育ったね。エンジニアの視点だけでなく、事業視点も持つ。それが重要だったんだよ」

湊は深く頷いた。確かに、1年前の自分と今の自分は大きく変わっていた。技術的なスキルだけでなく、ビジネス視点も持つようになっていた。それは、一夜にして変わったわけではない。少しずつ、対話を通じて積み重ねてきたものだった。

新たな挑戦への意欲

社内での講演依頼が増えていた。湊のチームの取り組みが、組織全体に影響を与えていた。

「湊さん、うちのチームでも同じ取り組みをしたいんです。教えてください」

他チームのリーダーからの依頼が相次いだ。

外部カンファレンスでの発表も行った。

「開発生産性という名の重圧から解放されるまで」

湊はブログ記事を執筆した。その記事が社内外で反響を呼んだ。

「この経験を、より多くの人に伝えたい」

湊は新しい役割を感じていた。「組織変革のファシリテーター」。それは技術者からビジネス視点も持つエンジニアリーダーへの成長だった。

循環する成長

田村が3ヶ月後、同じように後輩指導を始めていた。

「湊さん、後輩に教えることで、自分も成長できるんですね」

田村が嬉しそうに報告した。

「うん。知識や経験は、共有することで価値が増える。それが循環する成長なんだよ」

湊は実感していた。個人の成長だけでなく、チーム全体が成長する文化が確立されていた。

山田も湊を評価していた。

「湊君、君が種を蒔いた木が、大きく育ったね。まだまだ成長中だ。次の課題も見えてきただろう？」

「はい。まだまだ改善の余地があります。でも、チーム全体で考えれば、きっと道は開けると思います」

湊は希望を感じていた。エンドレスな改善への意欲が湧いてきた。

物語の終わりが、新しい始まりでもある。

解説：エンジニアリーダーとしての成長プロセス

湊が経験したように、エンジニアリーダーとして成長するには、技術的なスキルだけでなく、ビジネス視点も必要です。

エンジニアリーダーの役割

エンジニアリーダーは、技術的なスキルだけでなく、ビジネス視点も持つ必要があります。チームだけでなく、組織全体を俯瞰できる視点を獲得することで、より価値のある判断ができるようになります。

エンジニアリーダーとしての成長プロセス

技術的スキルの向上：

- 技術的なスキルを継続的に向上させる
- 新しい技術や手法を学び続ける

ビジネス視点の獲得：

- エンジニアの視点だけでなく、ビジネス視点も持つ
- なぜその機能が必要なのか、それがどうビジネスに貢献するのかを理解する

組織全体を俯瞰：

- チームだけでなく、組織全体を俯瞰できる視点を獲得
- 部門横断のテックリードとして、組織全体に影響を与える

メンターとしての成長：

- 後輩を育てることで、自分も成長する
- 知識や経験を共有することで、価値が増大する

循環する成長の実現

知識・経験の共有により、価値が増大します。個人の成長だけでなく、チーム全体が成長する文化を確立します。エンドレスな改善により、一度の成功で終わらず、継続的に改善を続けます。

DevOpsの成熟度がAI時代の成否を分ける

DORAレポート10年の変遷が示す核心的な結論は、AI駆動開発における成功は技術的な近道ではなく、過去10年間にわたるDevOpsの成熟度という歴史的な積み重ねの直接的な結果である、という事実です。生成AIの台頭はソフトウェア開発の前提を根底から揺るがしていますが、その恩恵を真に享受できる組織と、逆に混乱が増幅される組織との間には深刻な断絶が生じています。

「飛び箱」の比喩

この断絶を理解する鍵は、「飛び箱」の比喩にあります。CI/CD、アジャイル、そしてDevOpsといった能力を一段ずつ着実に積み上げてきた組織だけが、AI駆動開発という最も高い段を軽々と飛び越えることができるのです。

- まず、継続的インテグレーション（CI）という土台を築く
- その上に、アジャイルな働き方という次の段を置く
- さらにその上に、DevOpsの文化とプラクティスという段を積み上げる

このように能力を段階的に積み上げてきた組織だけが、AI駆動開発という最も高い段を軽々と飛び越えることができます。基礎ができていない組織が、いきなりAIという高い段に挑むことは無謀な挑戦であり、AIが「増幅器」として機能不全を拡大させる結果を招くだけです。

この歴史的な積み重ねの有無こそが、成果を出す組織と苦戦する組織の間に存在する深刻な断絶の根本原因なのです。

詳細な手法については、章末の「手法11：持続可能な開発成功事例集」を参照してください。

> ***出典:*** *DORA（DevOps Research and Assessment）レポート2025年版『State of AI-Assisted Software Development』、https://youtu.be/Dvo5Hhay-t0?list=TLGGO2hCbXy_mxozMTEyMjAyNQ*

手法11：持続可能な開発成功事例集

技術的負債返済と新規開発の両立

事例1：週1回のリファクタリング時間の確保

実施内容：

- 週1回、2時間のリファクタリング時間を確保
- 技術的負債のリスクスコアに基づいて優先順位を決定
- ボーイスカウトルールを実践（コードを触ったら、少しでも良くしてから離れる）

効果：

- 技術的負債が50%削減
- 開発速度が300%向上
- バグ発生率が70%減少

期間：

- 1ヶ月で制度確立、3ヶ月で効果実感、1年で大幅な改善

事例2：20%ルールの活用

実施内容：

- 開発時間の20%を技術的負債返済に充てる
- 技術的負債のリスクスコアに基づいて優先順位を決定
- 継続的に返済を進める

効果：

- 技術的負債が継続的に削減
- 開発速度が安定して向上
- チーム全体の技術力が向上

期間：

- 1ヶ月で制度確立、6ヶ月で効果実感

AIとの戦略的協働

事例3：技術的負債検出の自動化

実施内容：

- 毎週自動でコードベース全体をスキャン
- リスクスコアに基づく優先順位付け
- CI/CDパイプラインに組み込む

効果：

- 技術的負債の早期発見が可能に
- 継続的な監視により、技術的負債の蓄積を防ぐ
- 開発速度が5倍に向上

期間：

- 1ヶ月でシステム構築、3ヶ月で効果実感

事例4：AIペアプログラミングの導入

実施内容：

- AIがコードを提案し、人間がレビューして修正する

- 人間とAIの役割分担を明確にする
- 継続的に改善を進める

効果：

- 開発速度が5倍に向上
- 品質も保てる
- チーム全体のスキルが向上

期間：

- 1ヶ月で導入、3ヶ月で効果実感

組織文化の確立

事例5：多角的な評価軸の導入

実施内容：

- ビジネスインパクト、持続可能性、効率性の3つの軸で評価
- 数値だけでなく、定性的な評価も含める
- 定期的に評価軸を見直し、改善する

効果：

- 技術的負債返済が「必要な投資」として認識される
- リファクタリング時間が正式に確保される
- 組織全体の文化が変化

期間：

- 1ヶ月で評価軸設計、3ヶ月で制度確立、6ヶ月で文化定着

事例6：全社標準化への展開

実施内容：

- ノウハウを体系化
- 他チームに段階的に展開
- 定期的にフォローアップ

効果：

- 組織全体に良い影響を与える
- 他チームも同様の成果を上げる
- 組織全体の生産性が向上

期間：

- 3ヶ月でノウハウ体系化、6ヶ月で他チーム展開開始、1年で全社標準化

手法12：AI協働実践ガイド

AI活用の基本原則

1. AIは補助ツールであり、代替ではない

- AIが提案したコードをそのまま使うのではなく、人間がレビューして修正する
- AIが指摘した問題をそのまま信じるのではなく、人間が確認する
- AIの判断を盲信せず、人間の判断を優先する

2. AIと人間の役割分担を明確にする

AIが得意なこと：

- 大量のデータ分析
- パターン検出
- 網羅的なチェック

人間が得意なこと：

- コンテキスト理解
- 創意工夫
- 意思決定

3. 段階的に導入する

- 小さな実験から始める
- 効果が確認できたら、段階的に拡大する
- 失敗から学び、改善を続ける

AI活用の具体的な手法

技術的負債検出でのAI活用

手順：

1. AIツールにコードベースを分析させる
2. AIが指摘した問題箇所を人間が確認する
3. リスクスコアに基づいて優先順位を決定する
4. 継続的に監視する

効果：

- 網羅的な分析が可能
- 人間では見落としがちな問題を発見
- 時間の短縮

AIペアプログラミング

手順：

1. AIがコードを提案する
2. 人間がレビューして修正する
3. 継続的に改善を進める

効果：

- 開発速度が5倍に向上
- 品質も保てる
- 人間とAIの完璧な役割分担

テストケース生成でのAI活用

手順：

1. AIに仕様書を読み込ませる
2. AIがテストケースを提案する
3. 人間がレビューして、必要に応じて修正する

効果：

- 網羅的なテストケースの生成
- テストケース作成時間の短縮
- エッジケースの発見

AI活用のチェックリスト

導入前：

- [ ] AI活用の目的を明確にした
- [ ] チーム全体でAI活用の方針を合意した
- [ ] 小さな実験から始める計画を立てた

導入中：

- [ ] AIが提案したコードを人間がレビューしている
- [ ] AIの判断を盲信せず、人間の判断を優先している
- [ ] 失敗から学び、改善を続けている

導入後：

- [ ] AI活用の効果を測定している
- [ ] チーム全体でAI活用のスキルを向上させている
- [ ] 継続的に改善を続けている

章のまとめ

第7章では、湊が「開発生産性をあげるだけのエンジニアからの脱却」を果たし、技術的負債の返済と新規開発の両立、AIとの戦略的協働による開発速度5倍の実現、数値を超えた「価値創出」を評価する組織文化の確立を通じて、技術者からビジネス視点も持つエンジニアリーダーへ成長する過程を描きました。

主な学び

1. **技術的負債返済と新規開発の両立**: 時間配分の最適化、継続的な返済、価値の可視化により、両立が可能になる
2. **AIとの戦略的協働**: AIを補助ツールとして活用し、人間とAIの役割分担を明確にすることで、開発速度を大幅に向上させられる
3. **数値を超えた価値創出の評価**: 多角的な評価軸、定性的な評価、評価制度の設計により、組織文化を確立できる
4. **エンジニアリーダーとしての成長**: 技術的スキルの向上、ビジネス視点の獲得、組織全体を俯瞰、メンターとしての成長により、循環する成長が実現できる
5. **DevOpsの成熟度の重要性**: AI時代の成功は、DevOpsの歴史的積み重ねの上に成り立つ。基礎を着実に積み上げることが、AI駆動開発の成功につながる

DORAレポートが示す5つのアクション

これからの時代を生き抜くために組織は何をすべきか。DORAレポートは、以下の5つの具体的なアクションを提言しています。

1. AI導入を組織変革として扱う

AIを単なる効率化ツールとしてではなく、仕事の進め方や組織文化そのものを変える「**組織変革**」と位置づける。技術導入の先に、システム全体の変革を見据える必要があります。

2. 導入から「効果的な仕様」へ議論を移す

「Copilotを導入した」というプレスリリースを打って満足するのではなく、「AIがビジネスの成果に繋がっているか」を問い続けるべきです。議論の焦点を、導入の事実（Adoption）から効果的な仕様（Effective Use）へとシフトさせることが求められます。

3. デリバリーだけでなくウェルビーイングも評価する

AIは開発者の燃え尽きや業務上の摩擦を増大させるリスクをはらみます。デリバリーの速度や安定性に加え、従業員のウェルビーイングを重要な評価指標として組み込み、持続可能な開発環境を維持することが不可欠です。

4. プラットフォームエンジニアリングへの投資

開発者が価値創造に集中できる環境を整えるため、高品質な内部プラットフォームへの投資を強化します。優れた開発者体験（DevEx）は、AI時代の生産性の基盤となります。

5. バリューストリームマネジメント（VSM）の実践

局所的な改善に終始せず、アイデアの着想から顧客への価値提供まで、プロセス全体（バリューストリーム）を可視化し、システム全体の視点で改善を進めるアプローチがこれまで以上に重要になります。

これらのアクションは、AIという新たな波に乗り遅れまいと焦る必要はないことを示しています。真の近道は、DevOpsの原則に立ち返り、継続的デリバリー、小さなバッチでの作業、ユーザー中心の文化といった、地道な組織能力の向上に日々取り組むことです。それこそが、AI時代を勝ち抜くための唯一の、そして最も確実な道なのです。

> ***出典**: DORA（DevOps Research and Assessment）レポート2025年版『State of AI-Assisted Software Development』、https://youtu.be/Dvo5Hhay-t0?list=TLGGO2hCbXy_mxozMTEyMjAyNQ*

次章への展望

湊のチームは、1年間の取り組みにより驚異的な成果を上げました。開発速度300%向上、バグ発生率70%減少、技術的負債50%削減。これらの成果は、技術的負債返済と新規開発の両立、AIとの戦略的協働、数値を超えた価値創出の評価により実現されました。

湊は技術者からビジネス視点も持つエンジニアリーダーへ成長し、組織変革のファシリテーターとして活躍しています。おわりにでは、湊が執筆したブログ記事が社内外で反響を呼び、組織全体に変化の波が及ぶ希望が芽生える様子を描きます。