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 トップダウンの標準化では数値を追う組織になる

1年後の成果と組織文化の変化

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。

「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注目した。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、自動テストと、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」

「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、失敗した時、一人で抱え込んでいた自分。各職種の視点を学び、データ分析を始め、チーム全体で問題を共有し、AIを戦略的に活用し、自分たちで指標を作りながら自己成長する文化を確立した。そして今、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。

全社会議での事例発表。湊は1年間の取り組みを報告した。開発速度300%向上、バグ発生率70%減少、技術的負債50%削減。これらの数値が、組織全体に大きな反響を呼んだ。会議室には50名以上の社員が集まり、湊の発表に耳を傾けていた。

全社会議での発表をきっかけに、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。

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。

全社標準化への道と標準化をやめる決断

全社標準化の提案

数週間後、田中部長が湊を呼び出した。

「湊君、君のチームの取り組みは素晴らしい。これを全社標準にしたいと思うんだが、どうかな？」

湊は少し考えてから答えた。

「わかりました。進めてみます」

田中部長が続けた。

「それじゃあ、まずは標準化の指標を決めていこう。他チームのリーダーたちも集めて、一緒に決めてもらう」

湊は頷いた。

指標を決める会議

数日後、会議室に他チームのリーダーたちが集まった。湊は標準化の指標を決めるための議論を始めた。

「開発速度を何%向上させるか、バグ発生率を何%削減するか。こういった数値目標を設定しましょう」

他チームのリーダーが質問した。

「具体的には、どのくらいの数値が適切でしょうか？」

湊は資料を見ながら答えた。

「私たちのチームでは、開発速度300%向上、バグ発生率70%減少を実現しました。これを目標値として設定するのはどうでしょうか」

他チームのリーダーからの反応があった。

「うちのチームは現在、開発速度が年間10%低下しています。300%向上は難しいかもしれません」

「バグ発生率についても、現在の状況によって目標値は変わるべきでは？」

湊が追加の指標を提案した。

「それでは、週次で技術的負債返済のために確保されているリファクタリングやライブラリ更新の工数割合はどうでしょうか。私たちは週の工数の20%を確保しています」

「リファクタリング時間の確保率も指標にできます。週1回、2時間を確保することを標準化するのは？」

他チームのリーダーがFOUR Keysを提案した。

「DORAメトリクスのFOUR　Keysはどうでしょうか。デプロイ頻度、変更のリードタイム、変更失敗率、サービス復元時間。これらを標準化すれば、業界標準に沿った評価ができます」

湊が同意した。

「確かにFOUR　Keysは科学的に検証された指標です。デプロイ頻度を週1回以上、リードタイムを1週間以内、変更失敗率を15%以下、サービス復元時間を1時間以内。これらを目標値として設定するのはどうでしょうか」

他チームのリーダーからの質問があった。

「これらの指標を達成できなかった場合、どうなるのでしょうか？」

「評価に影響しますか？」

湊が答えた。

「評価に直結させることで、各チームが指標達成に向けて努力するようになります。四半期ごとに指標の達成率を確認し、達成できなかったチームには改善計画を立ててもらいます」

議論が進む中、湊は違和感を覚え始めた。数値目標を設定し、評価に直結させる。達成できなかったチームには改善計画を立ててもらう。この流れがどこかで見たことがある。

FOUR　Keysは科学的に検証された指標だが、一律の目標値を設定することが正しいのか。各チームの状況を無視した目標値は、かえって組織の成長を阻害するのではないか。

会議は一旦終了し、次回までに各チームの現状を調査することになった。

気づきの瞬間

湊は会議室を出て、自分のデスクに戻った。資料を整理しながら、1年前の自分たちの経験を思い出した。

田中部長から「開発生産性を向上させてくれ」と言われた。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。数値だけを追いかけて、本質的な改善ができなかった。形骸化した指標に依存し、本当に価値のあることを見失っていた。

FOUR Keysも導入したが、目標値を達成することに集中し、本質的な改善が後回しになった。デプロイ頻度を上げるために、小さな変更を頻繁にリリースしたが、価値のある変更が減った。リードタイムを短縮するために、テストを省略したが、バグが増えた。変更失敗率を下げるために、リリースを控えたが、価値提供の速度が落ちた。

湊は資料を見つめながら考えた。

「これって自分たちがやられていやだったことではないのか」

Four Keysという科学的に検証された指標も、一律の目標値を設定することで形骸化する。目標値を達成することが目的になってしまい、本質的な改善が後回しになる。外から開発生産性を求められても、ただ数値を追うばかりの組織になったり、開発生産性をあげる組織文化は育たない。標準化された指標を達成することが目的になってしまい、本質的な改善が後回しになる。各チームの状況を無視した一律の目標は、かえって組織の成長を阻害する。

指標そのものが悪いのではなく、指標を標準化し、評価に直結させることが問題なのだ。

標準化をやめて学びを広める活動へ

湊は田中部長に相談した。

「すみません、全社標準化をやめたいと思います」

田中部長が聞き返した。

「どうしてだ？」

湊が理由を説明した。

「指標を決めていく過程で気づいたのですが、これって1年前の私たちがやられていたことと同じなんです。外から数値目標を押し付けられても、組織文化は育ちません。代わりに、私たちの失敗や学びを広めていく活動に切り替えたいと思います」

田中部長は少し考えてから頷いた。

「なるほど。プロセスを標準化するのではなく、ノウハウや学びを共有するということか」

「はい。各チームが自分たちの状況に合わせて、プロセスと指標を定義できるよう支援します。私たちの経験を参考にしてもらい、それぞれのチームに最適な方法を見つけてもらう。それが持続可能な改善だと思います」

田中部長が答えた。

「わかった。それで進めよう。ノウハウや学びの展開を進めてほしい」

湊は頷いた。

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、それぞれのチームにあったプロセスと指標を定義していきます。私たちの経験を参考にしてもらい、自分たちの方法を見つけてもらいます」

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：トップダウンの標準化では数値を追う組織になる

ストーリーで描かれる「重圧」

数値目標の先の評価を問い直す段階で現れやすいのが、「数値を超えた価値創出をどう評価するか」という重圧です。湊のチームが経験したのも、ここから生じています。

- **AI活用から1年後の成果** 開発速度300%向上、バグ発生率70%減少、技術的負債50%削減。技術的負債返済の予算化、リファクタリング時間の制度化
- **「測れない価値」が「測れる価値」になった** という実感の共有
- **全社標準化の提案と指標を決める会議** 目標値の議論のなかで、1年前の自分たち（目標達成に集中し本質が後回しになった経験）との重なりを感じる
- **「これって自分たちがやられていやだったことではないか」** 外から開発生産性を求められてもただ数値を追うばかりの組織では、本質的な組織文化は育たないという気づき。指標の標準化と評価への直結が問題であることの認識。標準化をやめ、学びを広める活動へ切り替える

この重圧の背景には、**トップダウンで指標を標準化し評価に直結させると数値を追うだけの組織になり、本質的な組織文化は育たない構造**があります。数値を超えた価値創出を評価する文化を選んだ転換段階にあるのです。

開発生産性の本質的な定義

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。

開発生産性の本質は第1章で触れた通り、約束の信頼性（いつまでに何ができるかを語った約束がどれだけ信頼できるか）にあります。

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と事業責任者・PdMが見ているものの違いは第6章で触れた通りです。この違いを理解し、多角的な評価軸を設けることで、バランスの取れた評価ができます。

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、章末の持続可能な開発成功事例集のワークシートを参照してください。

7-2 湊の成長 自分たちで指標を作り、チームの生産性を考えるエンジニアリーダーへ

メンター湊の誕生

全社標準化の話から数ヶ月が経ち、新年度を迎えた。4月、新卒エンジニアの浜辺が配属された。浜辺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、ただコードを書くだけではなく、チームの生産性を考えられるようになろう。自分のコードがチーム全体にどう影響するのか。それを考えることが大事だよ」

浜辺は驚いた様子で聞き返した。

「チームの生産性...ですか？」

「うん。エンジニアは技術的なスキルだけじゃなく、チーム全体の生産性を考える視点が必要なんだ。自分たちの開発力がどのように事業やプロダクトに影響を与えているのか。それを理解することで、より価値のある開発ができるようになる。そういったスキルがこれからのAIエージェントを使いながら開発するエンジニア組織には必要になってくる。ただコードを書くだけならAIエージェントでもできるしね。」

湊は少し間を置いてから続けた。

「それから、開発生産性をお金に変換できる部分も理解する必要がある。開発生産性ツリーを使って、予算がどこに使われているのか、価値創出工数と価値維持工数を区別して考える。そうすることで、自分たちの開発が財務的にどう影響しているのかが分かるようになる」

浜辺が真剣な表情で聞いていた。

「そして、もう一つ重要なことがある。指標は周りから押し付けられるものではなく、自分たち自身で考えながら作っていくものなんだ。外から標準化された指標を押し付けられても、ただ数値を追うだけの組織になってしまう。自分たちの状況に合わせて、自分たちで指標を定義し、自己成長していく。それが大切なんだよ」

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」

浜辺が言った。

「うん。ただコードを書くだけではなく、チームの生産性を考え、自分たちの開発力が事業やプロダクトにどう影響するかを理解し、開発生産性をお金に変換できる視点を持ち、自分たちで指標を作りながら自己成長する。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

新卒の浜辺との対話で、湊は自身の成長を実感していた。

湊は振り返った。1年前、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。

その後、各職種の視点を学んだ。山田との対話を通じて技術的負債の重要性を知り、高橋とのランチで品質の視点を学び、飛鳥との議論で事業への影響を理解した。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。

そして、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、山田が「一人で抱え込む必要はない」と声をかけてくれた。

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。

その後、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。

そして、チームの生産性を自分たちで考えられるようになった。ただコードを書くだけではなく、チーム全体の生産性を考える視点を獲得した。自分の作業がチーム全体にどう影響するかを理解し、チーム全体で生産性を向上させる方法を考えるようになった。

自分たちの開発力が事業やプロダクトにどう影響を与えているかを理解できるようになった。開発の改善が事業価値にどうつながるかを考えるようになり、自分たちの開発力が顧客満足度や売上にどう影響しているかを意識するようになった。

開発生産性をお金に変換できるようになった。開発生産性ツリーを用いて、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理解できるようになった。開発生産性を財務的な言葉で説明できるようになり、経営会議でも、技術的な改善を財務的な成果として説明できるようになった。

そして、自分たちで指標を作り、自己成長する文化を確立した。外から標準化された指標を押し付けられるのではなく、自分たちの状況に合わせて指標を定義し、自己成長していく文化を確立した。指標は周りからではなく、自分たち自身で考えながら作っていくものだということを実践した。

そして今、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。ただコードを書くだけではなく、チームの生産性を考え、自分たちの開発力が事業やプロダクトにどう影響するかを理解し、開発生産性をお金に変換できる視点を持ち、自分たちで指標を作りながら自己成長する。それが重要だったんだよ」

湊は頷いた。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、チームの生産性を考え、自分たちの開発力が事業やプロダクトにどう影響するかを理解し、開発生産性をお金に変換できる視点を持ち、自分たちで指標を作りながら自己成長するようになっていた。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」。それは自分たちで指標を作り、チームの生産性を考えられるエンジニアリーダーへの成長だった。外から指標を押し付けられるのではなく、自分たちで考えながら指標を作る文化を広める。それが自分の役割だと感じていた。

循環する成長

浜辺が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」

浜辺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

ストーリーで描かれる「重圧」

「エンジニアリーダーとしての成長」が重圧に感じるなら、それは**循環する成長の段階**に立っている証かもしれません。湊が経験した重圧も、この構造から生じています。

- **メンターとして新卒に伝える**　チームの生産性を考え、開発力が事業・プロダクトにどう影響するかを理解し、開発生産性をお金に変換できる視点を持ち、自分たちで指標を作りながら自己成長することの重要性
- **社内講演・外部カンファレンス発表の増加**　社内外で経験を共有する機会が増える
- **知識や経験の共有で価値が増大する文化**　の実感。チーム全体が成長する文化の確立
- **技術者からリーダーへ、さらに組織変革のファシリテーターへ**　という成長の実感。自分たちで指標を作り、自己成長する文化を後輩に継いでいく

この重圧の背景には、**コードを書くだけでなく、生産性を考え・お金に変換し・指標を自分たちで作りながら成長するリーダー像が明確になる構造**があります。循環する成長を実感している段階にあるのです。

エンジニアリーダーの役割

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、チームの生産性を考え、自分たちの開発力が事業やプロダクトにどう影響するかを理解し、開発生産性をお金に変換できる視点を持ち、自分たちで指標を作りながら自己成長することが必要です。

エンジニアリーダーは、ただコードを書くだけではなく、以下の視点を持つ必要があります。

- **チームの生産性を考える視点**個人のコードだけでなく、チーム全体の生産性を考える
- **開発力と事業・プロダクトの関係を理解する視点**自分たちの開発力がどのように事業やプロダクトに影響を与えているのかを理解する

- **開発生産性をお金に変換できる視点**開発生産性ツリーを用いて、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理解する
- **自分たちで指標を作る視点**外から指標を押し付けられるのではなく、自分たちで考えながら指標を作り、自己成長する

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

チームの生産性を考える視点の獲得

- ただコードを書くだけではなく、チーム全体の生産性を考える
- 自分の作業がチーム全体にどう影響するかを理解する
- チーム全体で生産性を向上させる方法を考える

開発力と事業・プロダクトの関係を理解する視点

- 自分たちの開発力がどのように事業やプロダクトに影響を与えているのかを理解する
- 開発の改善が事業価値にどうつながるかを考える
- 自分たちの開発力が顧客満足度や売上にどう影響しているかを意識する

開発生産性をお金に変換できる視点

- 開発生産性ツリーを用いて、予算（財務）起点で開発投資を分解する
- 価値創出工数と価値維持工数を区別して理解する

- 開発生産性を財務的な言葉で説明できる
- 技術的な改善を財務的な成果として説明できる

自分たちで指標を作る視点

- 外から指標を押し付けられるのではなく、自分たちで考えながら指標を作る
- トップダウンの標準化では数値を追う組織になることを避ける
- 自分たちの状況に合わせて指標を定義し、自己成長する
- 指標は周りからではなく、自分たち自身で考えながら作っていくものだと理解する

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、章末の持続可能な開発成功事例集のワークシート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

章のまとめ

本章では、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、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、自分たちで指標を作り、チームの生産性を考える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. **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
2. **自分たちで指標を作る重要性**　外から指標を押し付けられるのではなく、自分たちで考えながら指標を作り、自己成長する
3. **エンジニアリーダーとしての成長**　チームの生産性を考え、自分たちの開発力が事業やプロダクトにどう影響するかを理解し、開発生産性をお金に変換で

きる視点を持ち、自分たちで指標を作りながら自己成長する

4. **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評価指標に組み込み、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求められます。

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。その実践が、AI時代を生き抜く道です。

> ***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*