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

経営会議への招待

AI活用を始めてから半年が経過した。湊のチームは着実に成果を上げていた。

開発速度は30%向上し、バグ検出率は40%向上していた。技術的負債の返済も進み、チーム全体の疲労度も軽減されていた。これまでのAI活用の取り組みは、開発生産性ツリーの観点から整理すると、戦略1（ValueStreamの高速化）によりIncrement（出口）を1.5倍にし、戦略2（Ops工数のAI置き換え）によりCapacity（入口）を1.5倍にすることで、全体として開発生産性を150%に向上させることができていた。

そんなある日、田中部長から湊に呼び出しがあった。

「湊君、これまでのAI活用の成果、経営層も注目してるよ。他部署からの評価も良好だったし、開発生産性ツリーの観点から説明できるという話も聞いた」

田中部長の表情には、期待と緊張が入り混じっていた。

「来月の経営会議で、成果発表の機会を設けてもらえることになった。湊君、君が発表してくれるか？」

湊は一瞬、返答に困った。経営会議で発表する。それは大きなチャンスかもしれない。でもエンジニアの視点だけで説明して、本当に理解してもらえるだろうか。

不安と期待が入り混じる

「エンジニアの視点だけで大丈夫かな？」

湊は正直に不安を口にした。

「君なら、経営層にも分かるように説明できるはずだ。でも準備はしっかりしてほしい。他部署の声も聞いてみるといいかもしれない」

田中部長の言葉に、湊は深く頷いた。確かにエンジニアの視点だけでは不十分かもしれない。これまでに他部署からの評価は得られていた。営業部からは顧客満足度の向上、サポート部からはバグ報告の削減、マーケティング部からはリリース計画の安定化という評価を得ていた。

でもそれをどう経営層に説明すればいいのか。技術的な話だけでは、伝わらないかもしれない。

伊藤部長との対話

湊は、田中部長の同席のもと事業責任者の伊藤部長との1on2ミーティングを設定した。

「湊君、君のチームの取り組みは素晴らしいと思う。でも正直、最初は『また技術的な話か』と思ってたんだ」

伊藤部長は率直に語った。

「事業視点では、どう見えますか？」

湊は勇気を出して質問した。

「うん。事業視点で見ると、いくつか評価できるポイントがある」

伊藤部長は資料を見ながら続けた。

「まず、顧客満足度の向上。バグが減ることで、お客様からのクレームが減った。これは直接的に事業価値につながっている」

「次に、開発コストの最適化。技術的負債の返済により、開発速度が向上した。同じ機能を開発するのに、以前より少ない工数で済むようになった。これはコスト削減につながる」

「最後に、市場投入スピードの安定化。リリース計画が安定することで、マーケティングや営業の計画も立てやすくなった。これは競争優位性につながる」

湊は驚いた。技術的な改善が、これほどまでに事業価値につながっているとは思っていなかった。

「技術的改善が事業価値につながってるんですね」

「そうだ。でも実は、もっと根本的なことがある」

伊藤部長は少し間を置いてから続けた。

「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ということだ。エンジニアが見ているものと、事業責任者が見ているものは違う。エンジニアはリードタイムやデプロイ頻度を見ている。でも事業責任者は、計画を信じて立てられるか、意思決定の前提として使えるかを見ている」

湊は真剣に聞き入った。

「今回の改善で、計画を信じて立てられるようになった。リリース計画が安定し、予測可能性が高まった。それが一番の価値だ。技術的な指標が改善しても、計画信頼性が向上しなければ、事業責任者には『良くなった実感』がない。でも今回は、その両方が実現できている」

「信頼性が、開発生産性の本質なんですね」

「そうだ。でもそれをどう説明するかが重要だ。エンジニアの視点だけで説明しても、経営層には伝わらない。事業視点で説明することで、初めて理解してもらえる」

伊藤部長の言葉に、湊は深く頷いた。確かにエンジニアの視点だけでなく、事業視点も必要だった。

事業視点の理解

湊は改めて考えてみた。開発生産性の本質は「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。」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）は異なる。Four Keysなどの指標は開発組織の内部安定性を測るが、事業計画の信頼性は直接測らない。これが、Four Keysが改善しているにもかかわらず、事業責任者・PdMには「良くなった実感」がないギャップの原因なのではないか。

解説：エンジニア視点とビジネス視点の融合

ストーリーで描かれる「重圧」

「経営層に開発生産性をどう説明するか」が重圧になるのは、**何が起きているかが関係者で共有されていない**ためです。湊が直面したのも、この構造から生じています。

- **経営層からの評価依頼**　「エンジニアの視点だけで大丈夫か」という不安。技術的な改善をどう伝えればよいか
- **これまでの成果の振り返り**　他部署からの評価を実感し、エンジニアの改善が会社全体に影響していることを改めて認識する

- **事業責任者との議論**　開発生産性の本質は約束の信頼性であること、開発組織が見ているものと事業責任者が見ているものは違うことを理解する
- **「計画を信じて立てられるようになったのが一番の価値だ」**　という評価。信頼性が開発生産性の本質だと腹落ちする

この重圧の背景には、**エンジニア視点だけでは伝わらず、事業視点での「信頼性」を共有して初めて評価が伝わる構造**があります。両視点の融合の必要性に気づいた段階にあるのです。

開発生産性の本質的な定義

湊が経験したように、エンジニア視点とビジネス視点の違いを理解し、融合させることで、共通言語を確立できます。

開発生産性の本質は第1章で触れた通り、約束の信頼性にある。

エンジニア視点とビジネス視点の違い

エンジニア視点

- 技術的品質、持続可能性、コードの複雑度
- 開発速度、バグ発生率、テストカバレッジ
- 技術的負債の管理、リファクタリングの重要性
- **測っているもの**　リードタイム、デプロイ頻度、変更失敗率、MTTR（内部状態・流動性・安定性）

ビジネス視点

- 売上貢献、コスト削減、市場投入速度
- 顧客満足度、競争優位性、事業成長
- ROI、投資対効果、財務的な成果

事業/経営視点

- 持続可能な資産とお金、財務的な成果
- ソフトウェア資産計上、減価償却費、P/Lへの影響
- 長期的な競争力、組織全体の成長
- **測っているもの**　計画信頼性・予測可能性（欲しいタイミングで欲しいものが出るか、計画は信じて立ててよいか）

測っているものが違う

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。これが、Four Keysが改善しているにもかかわらず、事業責任者・PdMには「良くなった実感」がないギャップの原因です。

なぜ視点の違いが問題になるのか

各レイヤーで「開発生産性」の意味が異なります。この違いを理解せずに議論すると、対立が生まれます。

- **時間軸の違い**　ビジネスは四半期単位、エンジニアリングは年単位での価値創出
- **価値の可視化困難** 技術的負債の影響は数ヶ月〜数年後に顕在化
- **専門性の壁** 技術的負債の深刻度を非技術者が理解するのが困難
- **測っている指標の違い**　開発組織は内部状態を測り、事業責任者・PdMは計画信頼性を測る

融合の重要性

相互理解を促進することで、より建設的な対話が可能になります。多角的な視点で判断することで、バランスの取れた意思決定ができます。エンジニアとビジネス側が協働することで、組織全体の生産性が向上します。

開発生産性ツリーの観点からの整理

開発生産性ツリーを用いることで、エンジニア視点とビジネス視点の融合を構造的に説明できます。

予算（財務）起点での開発投資の分解

- 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

この構造により、以下のことが明確になります。

- **価値創出工数と価値維持工数の区別**新規開発・エンハンス開発（価値創出）と保守開発・運用・管理業務（価値維持）を区別し、どこに時間が使われているかを可視化
- **財務と開発行動の接続**予算がどの施策に、どの工程で、どの開発フェーズで消費されているかを説明可能な形で提示
- **ROIの説明**投資対効果を財務言語で説明し、経営層と共通の理解を構築

開発生産性ツリーの観点から整理することで、技術的な改善がビジネス価値にどうつながるかを明確に示すことができます。

この「予算（財務）起点での開発投資の分解」という構造は、エンジニアと経営層が共通の財務言語で対話するための共通言語として機能します。予算から開発ライフサイクルまで一本で接続されたこの構造により、技術的な改善を財務的な成果として説明でき、経営層との相互理解を深めることができます。

6-2 PMやPdMと共に創る「共通の価値基準」

共通言語を探す

経営会議でのプレゼンに向けて、湊はチームメンバーとプレゼン内容を検討していた。

共通言語の模索

「技術用語をビジネス用語に翻訳する必要があるね」

山田が提案した。

「例えば、技術的負債をどう説明するか。経営層に『技術的負債』と言っても、理解してもらえないかもしれない」

湊はホワイトボードに書きながら考えた。

「技術的負債 → 開発効率の低下リスク。どうかな？」

「それだと、まだ抽象的かもしれない。もっと具体的に、コストや時間で説明できないか？」

飛鳥が加わった。

「リファクタリング → 将来への投資。これはどう？」

「いいね。でも投資対効果も示した方がいいかもしれない」

山田が補足した。

「コードの品質 → 継続的な競争優位性。これは事業視点で説明できそうだね」

山田が少し考えてから続けた。

「そういえば、以前のワークショップで学んだISO/IEC　25010の品質特性も、共通言語として使えるかもしれない。品質を8つの特性に分類することで、経営層にも多面的に説明できる。例えば、保守性の向上が開発速度の向上につながること、信頼性の向上が顧客満足度の向上につながることなど」

湊は頷いた。

「確かに。品質を単一の指標で測るのではなく、多面的に評価することで、経営層にも理解してもらいやすくなるかもしれない」

湊は書きながら続けた。

「技術的負債の返済により、開発速度が年間20-40%向上。これはコスト削減につながる」

「バグ修正コストをリリース後の30-100倍から開発時の10倍に削減。これも具体的な数値で示せる」

飛鳥が頷いた。

「これなら経営層にも伝わりそう。技術的な話を、事業価値に結びつけて説明する。それが重要だね」

山田も同意した。

「技術的な補足も必要だけど、まずは事業価値で説明する。その後に、技術的な詳細を説明する。その順序が重要だと思う」

湊はチームメンバーの意見を聞きながら、プレゼン資料を作成していった。

開発生産性ツリーの可視化

湊はホワイトボードの前に立ち、新しい図を描き始めた。

「実は、もう一つ説明したいことがあります。田中部長と話していたのですが開発投資がどこに使われているかを、構造的に見える化する方法です」

湊はホワイトボードに矢印と箱を描きながら続けた。

「予算から始まって、人件費に変換され、開発区分に配分されます。そして、工数、施策、バリューストリーム、開発ライフサイクルまで、一本の線でつながっています」

山田が興味深そうに見つめた。

「これは面白い。予算という財務の視点から、実際の開発行動まで接続できるんだね」

「はい。そして、ここが重要なんですが」

湊は開発区分の部分を指差した。

「開発区分は、価値創出工数と価値維持工数に分けられます。新規開発やエンハンス開発は価値創出。保守開発や運用、管理業務は価値維持です」

飛鳥が頷いた。

「これなら、どこにお金と時間が使われているか一目瞭然ですね。経営層にも『なぜ生産性が低いのか』が伝わるはずです」

「BigQueryとLookerで実際のデータを可視化できます。この構造で説明すれば、財務と開発行動が一本で接続され、説明可能な形で理解してもらえると思います」

山田が補足した。

「この構造を使えば、投資対効果も説明しやすくなる。予算がどの施策に、どの工程で、どの開発フェーズで消費されているかが明確になるからね」

湊はチームメンバー全員が納得しているのを確認し、プレゼン資料に開発生産性ツリーの図を追加した。

経営会議の前夜

経営会議当日。役員10名を前に、湊は緊張しながらも、準備してきた資料を説明し始めた。

経営会議での緊張のプレゼン

「開発生産性向上の取り組み報告」

湊はデータと事例を交えながら、20分間の発表を行った。

「まず、成果からご報告します。開発速度が30%向上し、バグ検出率が40%向上しました。これにより、顧客満足度が向上し、サポート対応の工数も削減できました」

経営層の反応は、最初は懐疑的だった。でも湊が具体的な数値と事例を示すにつれて、徐々に真剣な表情に変わっていった。

湊は画面を切り替え、開発生産性ツリーの図を映した。

「次に、開発投資がどこに使われているかを構造的に説明させていただきます。予算から始まって、人件費に変換され、開発区分に配分されます。そして、工数、施策、バリューストリーム、開発ライフサイクルまで、一本の線でつながっています」

湊は開発区分の部分を指差した。

「開発区分は、価値創出工数と価値維持工数に分けられます。新規開発やエンハンス開発は価値創出。保守開発や運用、管理業務は価値維持です。本来は価値創出工数に多く使うべきですが、現在は価値維持工数に時間を使っている状況でした」

湊はAI戦略の説明に移った。

「これまでのAI活用の取り組みは、開発生産性ツリーの観点から2つの戦略に整理できます。戦略1は、ValueStreamの高速化により、同じ投入工数でより多くの成果物を生み出すことで、Increment（出口）を1.5倍にします。戦略2は、Ops工数をAIに置き換えることで、余剰工数を新規開発に回し、Capacity（入口）を1.5倍にします。両戦略の組み合わせにより、開発生産性を150%に向上させることができます」

「技術的負債の返済により、開発速度が年間20-40%向上しています。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。開発コストの削減につながっています」

「バグ修正コストをリリース後の30-100倍から開発時の10倍に削減しました。これにより、バグ対応の工数が大幅に削減され、新機能開発に集中できるようになりました」

CEOが質問した。

「つまり短期的なスピードよりも、持続可能性が大事ということか」

「はい、長期的な事業成長のための基盤作りです。短期的には開発速度が少し低下するように見えるかもしれませんが、長期的には開発速度が向上し、コスト削減につながります」

湊は自信を持って答えた。

経営層との質疑応答

CFOが手を挙げた。湊は緊張しながら、CFOの方を向いた。

「投資対効果はどの程度？」

CFOの声は冷静だった。湊は深呼吸をして、準備してきた資料を見せながら答えた。

「開発コスト15%減、バグ対応工数40%減です。技術的負債の返済に投資した時間は全体の20%程度ですが、その投資により開発速度が年間20-40%向上しています。投資対効果は約2倍以上と試算しています」

湊は画面にグラフを映しながら続けた。

「具体的には、技術的負債の返済により、開発速度が年間20-40%向上。バグ修正コストをリリース後の30-100倍から開発時の10倍に削減。これにより、年間で約500万円のコスト削減が見込めます」

CFOは資料を見つめながら、少し考え込んだ様子だった。でも数値に納得したのか、小さく頷いた。

CTOが続けて質問した。

「他のチームにも展開できるの？」

CTOの声には期待が込められていた。湊は少し緊張しながらも、自信を持って答えた。

「まずノウハウを体系化して、段階的に展開する計画です。今回の取り組みで得た知見を他のチームにも共有していきます。具体的には、技術的負債の可視化方法、リファクタリング時間の確保方法、AI活用のベストプラクティスなどを文書化して、他チームに展開する予定です」

CTOは満足そうに頷いた。

「それは良い。ぜひ進めてほしい」

CEOも湊を見て言った。

「湊さん、素晴らしい取り組みだ。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重要だ。この取り組みを全社標準にしたい」

湊は予想以上の好反応に驚いた。経営層が技術的な改善の価値を理解してくれている。それはエンジニアの視点だけでなく、事業視点で説明したからだ。湊はこの時、ビジネス視点を持つことの重要性を実感した。技術的な改善を、ビジネス価値に変換して説明することで、経営層にも理解してもらえる。それが、エンジニアリーダーとしての成長だった。

解説：PMやPdMと共に創る「共通の価値基準」

ストーリーで描かれる「重圧」

データはあるが伝え方がわからない。そんな状態のとき、「共通の価値基準を創る」はとりわけ重くのしかかります。湊のチームが経験した重圧も同じ構造からです。

- **技術用語をビジネス用語に翻訳する作業**　技術的負債を開発効率の低下リスクとして、リファクタリングを将来への投資として説明する
- **開発生産性ツリーで予算から開発ライフサイクルまでの因果を可視化**　財務と開発行動を接続し、投資がどこでどう消費されているかを示す
- **経営会議での緊張のプレゼン**　データと事例に基づく説明で、技術的改善の価値を事業視点で伝える
- **CFOからの投資対効果の質問に数値で答える**　エンジニアとビジネス側の共通理解が深まる手応えを得る

この重圧の背景には、**専門用語と価値基準のズレを解消し、共通言語と可視化で初めて相互理解が進む構造**があります。共通の価値基準を創る実践段階にあるのです。

なぜ共通言語が必要なのか

湊が実践したように、共通言語を確立することで、エンジニアとビジネス側の相互理解が深まります。

専門用語による壁があります。技術用語を非技術者が理解するのが困難です。価値基準の不一致により、エンジニアとビジネス側で「価値」の定義が異なります。対話の不足により、定期的な対話機会がないと、相互理解が進みません。

共通言語の確立方法

技術用語の翻訳

- 技術的負債 → 開発効率の低下リスク
- リファクタリング → 将来への投資

- コードの品質 → 継続的な競争優位性

ISO/IEC 25010の品質特性を共通言語として活用

ワークショップで学んだISO/IEC　25010の品質特性は、経営層への説明でも共通言語として機能します。品質を8つの特性に分類することで、多面的に評価し、経営層にも理解してもらいやすくなります。

品質特性と事業価値の対応関係

- **保守性の向上** → 開発速度の向上、コスト削減
  - コードの可読性向上により、修正時間が短縮される
  - テストカバレッジ向上により、バグ修正コストが削減される
- **信頼性の向上** → 顧客満足度の向上、サポートコスト削減
  - ダウンタイムの削減により、顧客満足度が向上する
  - 障害発生時の自動復旧により、サポート対応時間が削減される
- **性能効率性の向上** → ユーザー体験の向上、インフラコスト削減
  - レスポンスタイムの短縮により、ユーザー体験が向上する
  - リソース利用効率の向上により、インフラコストが削減される
- **セキュリティの向上** → リスク管理、コンプライアンス対応
  - データの暗号化により、情報漏洩リスクが低減する
  - アクセス制御の強化により、コンプライアンス要件を満たす

このように、ISO/IEC　25010の品質特性を事業価値と対応付けることで、技術的な改善がビジネス成果にどうつながるかを明確に説明できます。

価値基準の統一

- 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

定期的な対話

- Tech & Business Syncミーティングなど、継続的な対話の場を設ける
- 月次で技術戦略を議論し、優先度を透明性高く決定する

ステークホルダー間の対話の重要性

定期的な対話により、各ステークホルダーの視点を理解できます。共通言語で対話することで、より建設的な意思決定ができます。エンジニアとビジネス側が協働することで、組織全体の生産性が向上します。

開発生産性ツリーを用いた財務と開発行動の接続

開発生産性ツリーを用いることで、財務（P/L・B/S）と開発行動を一本で接続し、共通言語として機能させることができます。

予算起点での可視化

- BigQueryとLookerを用いて実際のデータを可視化・分析
- 予算がどの施策に、どの工程で、どの開発フェーズで消費されているかを明確化

価値創出と価値維持の区別

- 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
- どこにお金と時間が使われているかを構造的に説明

ROI説明への活用

- 投資対効果を説明可能な形で提示

- なぜ生産性が低いのか、どこにお金と時間が使われているのかを構造的に説明

この方法により、エンジニアと経営層が共通の財務言語で対話でき、投資判断が適切に行えるようになります。

詳細な手法については、章末のステークホルダー対話ガイドのワークシートを参照してください。

6-3 工数・工期・事業KPIを包括した生産性の新しい評価軸

新しい評価制度の提案

経営会議の後、経営層から要請があった。

「この取り組みを制度化できないか？他のチームにも展開してほしい」

湊は山田と飛鳥と共に、新しい評価制度の設計を始めた。

会議室で、3人はホワイトボードの前に集まった。

「まず5名チーム全員からフィードバックを収集しよう」

湊はチームメンバーに意見を求めた。

「評価制度って、どういうものがいいと思う？」

佐藤が最初に発言した。

「工数や工期だけじゃなくて、品質も評価してほしい。バグが少ない方が結果的には効率的だと思うから。でも今の評価制度だと、バグが少なくても評価されない。むしろ、バグを出さないために時間をかけたら、評価が下がってしまう」

佐藤の言葉に、他のメンバーも頷いた。

山田も続けた。

「技術的負債の管理も評価してほしい。短期的な速度だけで評価されると技術的負債が蓄積してしまう。ワークショップで学んだように、形骸化した指標に依存すると、本質的な改善につながらない。リファクタリングの時間を確保しても、それが評価されないと、誰もやらなくなる」

飛鳥も意見を述べた。

「事業価値への貢献も評価してほしい。技術的な改善がどう事業に貢献しているかを可視化したい。エンジニアの視点だけでなく、PMや事業責任者の視点も取り入れた評価軸が必要だと思います」

湊はメンバーの意見を聞きながら、ホワイトボードに書き出していった。

「じゃあ、3つの軸で評価する制度を設計してみましょう。ビジネスインパクト、持続可能性、効率性。この3つで評価すれば、多角的な生産性評価ができると思います」

山田が頷いた。

「いいね。でもそれぞれの軸で、どういう項目を評価するのか、もっと具体的に決める必要があるね」

飛鳥も同意した。

「そうですね。例えば、ビジネスインパクトなら、顧客満足度の向上、売上への貢献、バグ対応工数の削減など。持続可能性なら、技術的負債の管理状況、コード品質の維持・向上、チームメンバーの成長など。効率性なら、開発速度、バグ発生率、テストカバレッジなど」

湊はメンバーの意見を聞きながら、3つの軸で評価する制度を設計した。

1. **ビジネスインパクト**（売上・顧客満足度への貢献）
    - 顧客満足度の向上
    - バグ対応工数の削減
    - 市場投入速度の安定化
2. **持続可能性**（技術的負債管理、チーム成長）
    - 技術的負債の管理状況
    - コード品質の維持・向上
    - チームメンバーの成長
3. **効率性**（開発速度、品質）
    - 開発速度の向上
    - バグ発生率の削減
    - テストカバレッジの向上

「全ステークホルダーが理解できる指標にしたい」

湊は設計した評価制度を説明した。

「エンジニアの視点だけでなく、PM、事業責任者の視点も取り入れた評価軸です。これにより、多角的な生産性評価が可能になります」

組織文化の変化の始まり

新しい評価制度の試験導入が始まった。

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。

田中部長が湊を呼び出した。

「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。

山田も湊を評価していた。

「湊さん、君が種を蒔いた木が、大きく育ってるね。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」

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。

「エンジニアだけの問題」から「組織全体の取り組み」へ。湊はその変化を実感していた。

解説：工数・工期・事業KPIを包括した生産性の新しい評価軸

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、「生産性の新しい評価軸を設計する」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。

- **ビジネスインパクト、持続可能性、効率性の3軸での新評価制度の提案** 単一指標や工数・工期だけでは全体像を把握できない限界を超える

- **開発生産性の本質（約束の信頼性）と内部品質投資の再定義を評価軸に組み込む**　技術的負債の返済を「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
- **工数・工期・事業KPIを包括した枠組みでROIを説明可能にする**　財務と開発行動を一本で接続する
- **「エンジニアだけの問題」から「組織全体の取り組み」へ** という変化の実感

この重圧の背景には、**単一指標や事業KPIとの断絶を解消し、多角的な評価軸で初めて適切な評価と説明ができる構造**があります。新しい評価軸の設計と組織文化の変化の始まりにあるのです。

開発生産性の再定義

湊が設計したように、多角的な生産性評価の枠組みを構築することで、より適切な評価が可能になります。

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」です。この信頼性を評価するためには、多角的な視点が必要です。

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。

本質的な問題

遅れること自体が問題なのではない。「なぜ遅れたのか」「次はどう変わるのか」を説明できないことが問題です。これは能力不足ではなく、構造的に説明可能な情報が欠けている状態なのです。

なぜ多角的な評価が必要なのか

単一指標の限界があります。工数や工期だけでは、開発生産性の全体像を把握できません。事業KPIとの連動不足により、技術的な指標と事業KPIが連動していません。多角的な評価の必要性から、ビジネスインパクト、持続可能性、効率性を包括的に評価する必要があります。

多角的な生産性評価の枠組み

ビジネスインパクト

- 売上・顧客満足度への貢献
- 市場投入速度
- コスト削減効果

持続可能性

- 技術的負債管理
- チーム成長
- コード品質の維持・向上

効率性

- 開発速度
- 品質（バグ発生率、テストカバレッジ）
- コスト効率

開発生産性ツリーを用いた評価方法

開発生産性ツリーを用いることで、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できます。

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

この構造により、以下のことが明確になります。

- **価値創出工数と価値維持工数の区別**
  - 価値創出工数　新規開発、エンハンス開発（将来の事業価値・収益・競争優位を生み出す）
  - 価値維持工数　保守開発、運用、管理業務（既存資産を「使い続けられる状態」に保つ）
- **財務（P/L・B/S）と開発行動の接続**
  - 予算がどの施策に、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
  - BigQueryとLookerを用いて実際のデータを可視化・分析
  - ROIを説明可能な形で提示

AIによる開発生産性向上戦略（開発生産性ツリーのセクション8）

開発生産性ツリーのセクション8では、AIを活用した開発生産性向上の戦略を定義しています。

目的

- Capacity（投入工数）に対して、Increment（成果物）の量を現状100%から150%にする

戦略1:　ValueStream（工数・工期）をAIネイティブなプロセスで高速化し、Increment（出口）を1.5倍にする

ValueStreamの各工程をAIネイティブなプロセスで高速化し、同じ投入工数でより多くの成果物を生み出す戦略です。

- **「問題を定義するところ」**

    - 要求定義をもとに要件定義を詰める部分
    - 人同士が泥臭く合意するところ
    - 意思決定を加速するためにAIを使う
    - **役割** 人がPilot、AIがCopilot
        - 人間が意思決定の主導権を持ち、AIが情報整理や選択肢の提示を支援
        - 要件定義の効率化により、リードタイムを短縮
- **「問題を解くところ」**
    - 要件定義をもとにタスクを分解し実装からリリースまでを行う部分
    - 作業はAI。品質は人が担保する
    - **役割** 人がCopilot、AIがPilot
        - AIがタスク分解、実装、テストケース生成など、実行作業をAIが主導
        - 人間が品質チェックや承認を行う
        - 実装からリリースまでの工程を高速化

効果 ValueStream全体のリードタイム短縮により、Increment（出口）を1.5倍にする。

戦略2:　Opsの工数をAIに置き換え(or協働)し余剰工数を25人月分新規に回して、Capacity（入口）を1.5倍にする

開発区分における（保守開発～管理業務）までの部分の工数をAIに置き換えることで、新規開発に回せる工数を増やす戦略です。

- **工数分析とAI置き換えプロセス**

1. **工数分析**　開発区分における（保守開発・運用・管理業務）までの部分の工数を分析する

        - （例）全体の40%の工数をかけていた場合、AIによる改善インパクトがある部分をプロセスから探す

    2. **AI置き換えの実施** AIに置き換え可能な業務を特定し、AI Agentを導入
    3. **効果測定** 置き換えた工数を定量化し、新規開発への転換効果を測定

- **AI Agentの具体例**

    - **保守開発** ポストモーテムを書くAI Agent、障害分析をして事前に対策をしてくれるAI、コードレビュー支援AI
    - **運用** 障害対応の自動化AI、ログ分析・異常検知AI、問い合わせ対応AI
    - **管理業務** メンバーの目標設定から目標トラッキング、評価をサポートしてくれるAI Agent、会議議事録作成AI、予算策定支援AI

効果　Ops工数の削減により、余剰工数25人月分を新規開発に回し、Capacity（入口）を1.5倍にする。

Capacity（投入工数）とIncrement（成果物）の関係

開発生産性ツリーの観点から見ると、AI戦略は以下の関係で整理できます。

- **Capacity（投入工数）** 開発に投入される工数（人月）
- **Increment（成果物）** 開発によって生み出される成果物の量

目標 Capacity（投入工数）に対して、Increment（成果物）の量を現状100%から150%にする。

戦略の組み合わせ

- **戦略1** Increment（出口）を1.5倍にする → 同じ投入工数でより多くの成果物を生み出す
- **戦略2** Capacity（入口）を1.5倍にする → Ops工数を削減して新規開発に回す工数を増やす
- **両戦略の組み合わせ** 開発生産性を150%に向上させる

このように、開発生産性ツリーのセクション8を用いることで、AIへの投資が開発生産性の向上にどうつながるかを明確に説明できます。経営層にも理解してもらいやすい形で、AI戦略の効果を提示することが可能になります。

- 予算がどの施策に、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
- BigQueryとLookerを用いて実際のデータを可視化・分析
- **ROI説明への活用**
  - 投資対効果を説明可能な形で提示
  - なぜ生産性が低いのか、どこにお金と時間が使われているのかを構造的に説明

この方法により、エンジニアと経営層が共通の財務言語で対話でき、投資判断が適切に行えるようになります。

評価軸の設計方法

各軸の評価項目を明確にします。定量的な指標と定性的な評価を組み合わせます。定期的に評価軸を見直し、改善します。

DORA Metricsとビジネス指標のマッピング

多角的な生産性評価において、DORA Metrics（Four Keys）をビジネス指標とマッピングすることで、エンジニア視点とビジネス視点の橋渡しが可能になります。DORA Metrics（Four Keys）は、技術的な指標でありながら、ビジネス価値に直結するため、両者の間の橋渡しとして機能します。

DORA Metrics → ビジネス指標のマッピング表

DORA Metrics	ビジネス指標	説明
デプロイ頻度	市場投入速度	デプロイ頻度が高いほど、市場への対応速度が上がり、競争優位性が高まる
変更のリードタイム	開発コスト削減	リードタイムが短いほど、開発コストが削減され、ROIが向上する
変更失敗率	顧客満足度・サポートコスト	失敗率が低いほど、顧客満足度が向上し、サポートコストが削減される
サービス復元時間	事業継続性・信頼性	復旧時間が短いほど、事業への影響を最小化し、信頼性が向上する

2025年版レポートの8つの新たな測定要因

2025年版DORAレポートでは、従来のFour Keysに加えて、以下の8つの新たな測定要因が用いられています。これらは、従来の「スループット」や「安定性」に加え、人的側面やビジネス価値までを包含する、より包括的な評価軸となっています。

増やしたい項目

- **チームパフォーマンス** チームとしての目標達成度
- **プロダクトパフォーマンス** 製品がもたらすビジネス価値
- **デリバリースループット** ソフトウェア提供の速度
- **個人の有用性** 個人の生産性や貢献実感
- **価値ある仕事** 意義や手応えを感じる業務

減らしたい項目

- **デリバリーの不安定性** リリースに伴う問題の発生度

- **摩擦（フリクション）** 業務を妨げる障害や非効率
- **燃え尽き症候群（バーンアウト）** 従業員の心身の疲弊度

2025年版レポートの8つの測定要因との統合

これらの8つの新たな測定要因を、3つの評価軸に統合することで、より包括的な評価が可能になります。

- **ビジネスインパクト** プロダクトパフォーマンス、デリバリースループット
- **持続可能性**　チームパフォーマンス、個人の有用性、価値ある仕事、燃え尽き症候群（低いほど良い）
- **効率性** デリバリーの不安定性（低いほど良い）、摩擦（低いほど良い）

これらの指標を組み合わせることで、エンジニア視点とビジネス視点の両方を包含した、包括的な生産性評価が可能になります。技術的な改善がビジネス価値にどうつながるかを明確に示すことができます。

詳細な手法については、章末の多角的生産性評価マトリクス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9 多角的生産性評価マトリクス

評価軸の設計

多角的な生産性評価を行うためには、3つの軸で評価すると、バランスの取れた判断ができます。

1. ビジネスインパクト

評価項目

- 顧客満足度の向上（NPS、顧客満足度調査）
- 売上への貢献（機能別売上分析）
- バグ対応工数の削減（サポート対応時間の削減）
- 市場投入速度の安定化（リリース計画の遵守率）

測定方法

- 定量的指標 NPS、売上データ、サポート対応時間
- 定性的評価 顧客フィードバック、営業・サポートからの評価

2. 持続可能性

評価項目

- 技術的負債の管理状況（技術的負債スコア、複雑度）
- コード品質の維持・向上（テストカバレッジ、コードレビュー）
- チームメンバーの成長（スキルマップ、学習時間）
- ドキュメント整備状況（ドキュメントカバレッジ）

測定方法

- 定量的指標 技術的負債スコア、テストカバレッジ、スキルマップ
- 定性的評価 コードレビュー、チームメンバーの成長実感

3. 効率性

評価項目

- 開発速度（デプロイ頻度、変更のリードタイム）
- 品質（バグ発生率、変更失敗率）
- コスト効率（開発コスト、バグ修正コスト）

測定方法

- 定量的指標 デプロイ頻度、リードタイム、バグ発生率
- 定性的評価 チームの疲労度、開発のしやすさ

評価の実施方法

ステップ1 現状の把握

各軸の評価項目について、現状を把握します。定量的な指標と定性的な評価を組み合わせて、総合的に評価します。

ステップ2 目標の設定

各軸について、具体的な目標を設定します。目標は、SMART（Specific, Measurable, Achievable, Relevant, Time-bound）の原則に従って設定します。

ステップ3 定期的なレビュー

四半期ごとに評価を実施し、進捗を確認します。必要に応じて、目標や評価項目を見直します。

DORA Metricsの具体的な活用方法

多角的な生産性評価において、DORA Metricsを活用する具体的な方法を以下に示します。

DORA Metricsの測定方法

1. デプロイ頻度（Deployment Frequency）

- **測定** 本番環境へのリリース回数を期間で割る（例）週1回、月1回
- **目標設定**　エリートレベル（1日1回以上）、高パフォーマンス（週1回〜月1回）
- **ビジネス指標との関連** 市場投入速度、競争優位性

2. 変更のリードタイム（Lead Time for Changes）

- **測定** コードコミットから本番デプロイまでの時間を測定
- **目標設定** エリートレベル（1時間未満）、高パフォーマンス（1日〜1週間）
- **ビジネス指標との関連** 開発コスト削減、ROI向上

3. 変更失敗率（Change Failure Rate）

- **測定** 本番デプロイ後に障害が発生した回数を全デプロイ回数で割る
- **目標設定** エリートレベル（0-15%）、高パフォーマンス（16-30%）
- **ビジネス指標との関連** 顧客満足度、サポートコスト削減

4. サービス復元時間（Time to Restore Service）

- **測定** 本番障害の発生から復旧までの時間を測定
- **目標設定** エリートレベル（1時間未満）、高パフォーマンス（1日未満）
- **ビジネス指標との関連** 事業継続性、信頼性向上

DORA Metricsとビジネス指標の統合レポート

定期的にDORA Metricsとビジネス指標を統合したレポートを作成することで、技術的な改善がビジネス価値にどうつながるかを明確に示すことができます。

レポート例

- 「デプロイ頻度が週1回から1日1回に向上 → 市場投入速度が30%向上 → 売上に10%の貢献」
- 「変更失敗率が20%から10%に削減 → サポート対応時間が40%削減 → コスト削減500万円/年」

このような形で、DORA Metricsをビジネス指標と関連付けることで、エンジニア視点とビジネス視点の橋渡しが可能になります。

> ***出典*** *DORA（DevOps Research and Assessment）レポート、『Accelerate: The Science of Lean Software and DevOps』*

評価結果の活用

評価結果は、以下の目的で活用します。

- **チームの成長** 評価結果を基に、チームの強みと弱みを把握し、改善計画を立てる
- **意思決定の支援** 評価結果を基に、技術投資の優先順位を決定する
- **組織全体の改善** 評価結果を共有し、組織全体の改善に活用する

手法10 ステークホルダー対話ガイド

対話の目的

エンジニアとビジネス側の対話を促進することで、相互理解を深め、より建設的な意思決定ができるようになります。

対話の準備

ステップ1 相手の視点を理解する

対話の前に、相手の立場や関心事を理解します。エンジニアの視点とビジネス視点の違いを認識し、共通言語を準備します。

ステップ2 目的を明確にする

対話の目的を明確にします。技術的な課題を説明するのか、投資判断を求めるのか、優先順位を決定するのか。目的に応じて、準備する資料や説明方法を変えます。

ステップ3 資料の準備

技術的な内容を、ビジネス用語で説明できる資料を準備します。具体的な数値や事例を含め、視覚的に分かりやすい資料を作成します。

対話の進め方

1. 共通言語の確立

技術用語をビジネス用語に翻訳します。

- 技術的負債 → 開発効率の低下リスク
- リファクタリング → 将来への投資
- コードの品質 → 継続的な競争優位性

2. 事業価値での説明

技術的な改善を、事業価値に結びつけて説明します。

- 開発速度の向上 → コスト削減、市場投入速度の向上
- バグ発生率の削減 → 顧客満足度の向上、サポート対応工数の削減
- 技術的負債の返済 → 長期的な開発速度の向上、コスト削減

3. 具体的な数値での説明

定性的な説明だけでなく、具体的な数値で説明します。

- 開発速度が30%向上 → 開発コスト15%削減
- バグ検出率が40%向上 → バグ対応工数40%削減
- 技術的負債の返済 → 開発速度が年間20-40%向上

定期的な対話の場

Tech & Business Syncミーティング

月次で実施し、エンジニアとビジネス側が技術戦略を議論します。優先度を透明性高く決定し、相互理解を深めます。

四半期レビュー

四半期ごとに、開発生産性の状況を全レイヤーで共有します。評価結果を基に、改善計画を立てます。

投資判断の透明化

開発生産性向上への投資判断を透明にし、全ステークホルダーが納得できる判断をします。

対話の効果

定期的な対話により、以下の効果が得られます。

- **相互理解の促進** 各ステークホルダーの視点を理解できる
- **意思決定の改善** 共通言語で対話することで、より建設的な意思決定ができる
- **組織全体の成長**　エンジニアとビジネス側が協働することで、組織全体の生産

性が向上する

章のまとめ

本章では、湊がエンジニアの視点だけでなく、PM、PdM、事業責任者の視点も取り入れた新しい生産性評価の枠組みを提案する過程を描きました。

主な学び

1. **エンジニア視点とビジネス視点の融合**　各レイヤーで「開発生産性」の意味が異なることを理解し、共通言語を確立する
2. **共通の価値基準の確立**　技術用語をビジネス用語に翻訳し、事業価値に結びつけて説明する
3. **多角的な生産性評価** 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
4. **ステークホルダー間の対話**　定期的な対話により、相互理解を深め、より建設的な意思決定ができる

次章への展望

本章で新しい生産性評価の枠組みを描いた。次の章では、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く。