● 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

午前2時。オフィスのフロアで明かりがついているのは湊のデスクだけだった。

障害対応が一段落し、チームメンバーはようやく帰宅した。しかし湊は一人残っている。デスクの上には資料が広がっている。先日の失敗を振り返る資料、過去のプロジェクトデータ、そして今日の会議のメモ。

湊はコーヒーカップを手に取り、冷めたコーヒーを一口飲んだ。苦い。

「開発生産性って、結局何なんだろう？」

先日田中部長から言われた言葉が頭に浮かぶ。「開発生産性を向上させてほしい」「数字を出せ」。でも、その「数字」とは何なのか。コード行数なのか、リリース数なのか、それとも別の何かなのか。

湊はノートPCを開き検索バーに打ち込んだ。

「開発生産性 指標」
「生産性向上 エンジニア」

検索結果には様々な記事が並ぶ。湊は一つ一つクリックして読んでいく。

ある記事は「コード行数で測るべき」と主張している。1日あたりのコード行数、1週間あたりのコード行数。でも、湊は首を横に振った。コード行数だけでは、無駄なコードを書けば数字は上がる。リファクタリングでコードを削減すれば数字は下がる。これが生産性の指標として適切なのか。

別の記事は「機能リリース数が重要」と書いている。月に何個の機能をリリースしたか。でも、今回の失敗を思い出す。急いで機能をリリースした結果、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。

さらに別の記事では「ストーリーポイント」や「ベロシティ」といった言葉が出てくる。アジャイル開発の指標だ。でも、これも完璧ではない。ストーリーポイントは見積もりの精度に依存するし、ベロシティはチームの成熟度によって変わる。

「どれも一理あるけど、どれも完璧じゃない...」

検索結果に並ぶ指標の数々

湊は画面を見つめた。検索結果には他にも様々な指標が並んでいる。

「デプロイ頻度」「変更のリードタイム」「変更失敗率」「回復時間」。DORAメトリクスという言葉も出てくる。でも、これらも技術的な指標に偏っている気がする。

さらに別の記事では「物的生産性」という言葉が出てくる。投入量に対する産出量の比率。労働時間あたりのコード行数、機能リリース数など。数値は出しやすい。でも、品質や価値は測れない。無駄なコードを書いても数字は上がる。

そして「付加価値生産性」という記事もある。ビジネス価値やユーザー価値を生み出す効率性。これは理想的な指標だと思う。でも、どう測るのか。ユーザー価値は数値化しにくい。ビジネス価値も、すぐには見えない。

「コード行数？機能リリース数？でも、今回の失敗を考えると…」

これは「生産性が高い」と言えるのだろうか？

確かに機能はリリースできた。でも、その後の障害対応でチームは疲弊した。ユーザーからの信頼も失った。長期的に見ると、これは生産性が高いと言えるのだろうか。

湊はノートに書き出し始めた。

生産性の指標として考えられるもの

- コード行数 → 無駄なコードを書けば上がる
- 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 → 見積もりの精度に依存
- デプロイ頻度 → 技術的な指標のみ
- 変更のリードタイム → 技術的な指標のみ
- 物的生産性 → 数値は出しやすいが、品質や価値は測れない
- 付加価値生産性 → 理想的な指標だが、どう測るのかが難しい

どれも一面的だ。もっと多角的な視点が必要なのではないか。

窓の外はまだ暗い。湊はコーヒーを淹れに行きデスクに戻った。オフィスには誰もいない。静まり返った空間で、湊は改めて考えた。

こういう根本的な問いに向き合うのは初めてだった。これまで湊は「与えられたタスクをこなす」ことに集中していた。でも、リーダーになって3ヶ月。もっと大きな視点で物事を考える必要があると感じていた。

山田との朝の対話

翌朝8時。湊は早めに出社した。昨日の深夜、様々な指標について考えていたが、答えは見つからなかった。もっと経験のある人に聞いてみたい。そう思い、山田に相談しようと決めていた。

エレベーターでの偶然の出会い

エレベーターホールで湊は山田と偶然一緒になった。

「おはよう、湊。昨日はお疲れさま。まだ早い時間だね」

山田はいつも冷静だ。

「おはようございます。山田さん、昨日のことすみませんでした」

「湊のせいじゃないよ。構造的な問題だと思う。急いでリリースした結果、バグが出た。でも、その背景には仕様の曖昧さやスケジュールの圧力があった。一人の責任じゃない」

エレベーターが1階に到着し、二人はオフィスのカフェコーナーへ向かった。まだ誰もいない静かな空間だ。自動販売機でコーヒーを買い立ち話が始まる。

「山田さん、実は昨日、開発生産性について考えていたんです。コード行数とか機能リリース数とか、いろんな指標があるけど、どれも一面的な気がして...」

湊の質問に山田は少し考えてから答えた。

「難しい質問だね。でも、技術的負債を返済する時間も生産性の一部だと思うんだよね」

「技術的負債...ですか」

「うん。過去のデータを見ると、技術的負債の蓄積で開発速度が年間20-40%低下してる。でも、それをどう上司に説明すればいいのか、僕も悩んでるんだ」

山田は自分のノートPCを開きチームの開発速度の推移グラフを見せた。

「これ、去年と今年の比較。同じ規模の機能開発に、今は1.3倍以上の時間がかかってる。コードベースが複雑になって、どこを触っても影響範囲が広がるからね」

湊は画面を見つめた。グラフには明確な下降トレンドが示されている。

去年の同じ時期、ある機能の開発には平均して5日かかっていた。でも、今年は7日かかる。同じ規模の機能なのに、開発時間が1.4倍になっている。

「これ、部長に見せたことあるんですか？」

「何度か。でも、『スピードが落ちてる』としか受け取られない。なぜ遅くなったのか、技術的負債の返済にどれだけ時間がかかるのかは、なかなか理解してもらえないんだ」

山田はコーヒーを一口飲んだ。

「例えば、先月のプロジェクト管理モジュールの改修。本来なら3日で終わるはずだった。でも、既存コードが複雑すぎて、影響範囲の調査に1日かかった。テストがないから、変更がどこまで影響するかわからない。ドメインモデルも曖昧で、モジュール間の依存関係が把握しにくい。リファクタリングに2日、実装に3日かかった。合計6日。本来の2倍だ」

「技術的負債の返済に時間をかけるべきだったんですか？」

「そうだね。でも、その時間を確保するのが難しい。新機能の開発が優先されるから。結果として、技術的負債は蓄積し続ける。そして開発速度はさらに低下する。負のスパイラルだ」

「他の人の意見も聞いてみたら？特に高橋さんとか」

「高橋さん...ですか」

「うん。品質の視点から見た生産性って、僕たちエンジニアとは違うと思うんだ。湊がリーダーとしていろんな視点を理解するのは大事だよ」

山田の助言は的確だった。湊は改めて、生産性という言葉が持つ多面性を感じていた。

「ありがとうございます。ランチの時にでも、高橋さんに聞いてみます」

QA部の高橋とのランチミーティング

12時。社内カフェテリアは混み合っている。湊は高橋を見つけ声をかけた。

「高橋さん、少しお時間いいですか？」

「湊さん、どうしました？」

高橋はトレイを持って立っていた。湊もトレイを持ち、二人は空いている席に座った。

「実は、開発生産性について考えていて。山田さんから、高橋さんの意見も聞いてみたらって言われて...」

「開発生産性ですか」

高橋は箸を置き少し考えてから答えた。

「正直、最近は品質を犠牲にした速度重視が多くて...」

「というと？」

「テスト工数がスケジュールに入らないんです。『これ、いつリリースするんですか？』って聞くと、『来週』とか言われて。でも、十分なテストをする時間がない」

高橋の声には責任感が滲んでいた。

「例えば、先月の経費精算モジュールの新機能。リリース2週間前にテスト依頼が来た。でも、テストケースを作成するのに1週間、実際のテストに1週間かかる。合計2週間。でも、開発側は『1週間でテストしてほしい』って言うんです」

「それは...」

「無理です。テストを急いでやると、見落としが出る。結果として、リリース後にバグが見つかる。そして、そのバグ修正にまた時間がかかる」

高橋はスマートフォンを取り出し、保存していた資料を見せた。

「バグが多いと、結局後で工数が10倍かかるんですよ。業界データでは、バグ修正コストは段階的に増大するんです」

「段階的に？」

「要件定義段階は1倍、実装段階は10倍、リリース後は30-100倍。昨日の障害対応、どれだけの工数がかかりましたか？」

湊は言葉に詰まった。確かにチーム全員で徹夜して対応した。もし開発中に見つけていれば、1人が数時間で修正できたはずだ。

「昨日の障害、もし開発中にテストで見つけていれば、1時間で修正できたはずです。でも、リリース後だったから、チーム全員で徹夜対応。ユーザーへの影響も大きかった。これって、本当に生産性が高いと言えるんでしょうか？」

高橋の言葉は、湊の中で何かが変わるきっかけになった。

速度だけじゃダメなんだ...

「品質も生産性のうちだと思うんですけどね。バグが少なければ、後で修正する時間がかからない。ユーザーからの信頼も得られる。長期的に見ると、品質を重視した方が生産性が高いはずなんです」

「でも、スケジュールの圧力があると...」

「わかります。でも、その圧力が結果として品質を下げ、長期的な生産性を下げている。負のスパイラルです」

高橋は箸を手に取り、ご飯を一口食べた。

「QA部として、私たちは品質を守る責任がある。でも、その時間が確保されない。これが続くと、チーム全体の生産性が下がるんです」

QA部という別組織からの視点は、湊にとって新鮮だった。エンジニアの視点とは違う、品質という軸での生産性の捉え方。これも欠かせない視点だ。

会議室への足取り

午後5時。会議室C-2で湊は飛鳥との1on1ミーティングに臨んでいた。

飛鳥との夕方の対話

「湊君、昨日は大変だったね。でも、ビジネス的にはリリースできて良かったよ」

「...良かった、ですか？」

湊の声には疑問が滲んでいた。バグが発生し、チームは徹夜で対応した。ユーザーからのクレームも殺到した。これが「良かった」と言えるのか。

「いや、もちろんバグは良くないけど。でも、期日通りにリリースしたことで、営業チームが顧客に説明できたし、売上の目標も達成できそうなんだ」

飛鳥はノートPCの画面を湊に見せた。売上予測のグラフが表示されている。

今月の売上目標は1000万円。今回リリースした機能によって、新規顧客の獲得が見込まれている。グラフには、リリース後の売上予測が示されていた。

「でも、これって本当に生産性が高いと言えるんでしょうか？」

湊の質問に飛鳥は画面から目を上げた。

「...実は私も悩んでる。短期的な売上は大事だけど」

飛鳥は少し声を落とした。

「PMとしては、ユーザー価値の創出が一番大事だと思ってる。市場への迅速な対応も必要。でも、持続可能性も大事なんだよね」

「持続可能性...」

「うん。今回みたいなトラブルが続くと、ユーザーからの信頼を失う。サポートチームも疲弊する。長期的に見ると、ビジネスにとってマイナスなんだ」

飛鳥はコーヒーを一口飲んだ。

「例えば、先月リリースした機能。リリース直後は売上が上がった。でも、バグが多くてユーザーからのクレームが続いた。結果として、解約が増えた。長期的にはマイナスだった」

「それでも、スピードを優先するんですか？」

「それが難しいところなんだ。競合他社が似た機能を出す前にリリースしないと、市場を取られる。でも、品質を犠牲にすると、長期的にはマイナスになる。バランスが難しい」

飛鳥は画面を閉じ、湊を見た。

「湊君はどう思う？生産性って何だと思う？」

湊は少し考えてから答えた。

「正直、わかりません。コード行数でもないし、機能リリース数でもない。でも、何なのか...」

「私も同じことを考えてる。PMとして、機能リリース数は重要だ。でも、それだけじゃない。ユーザーが本当に価値を感じてくれるか。長期的に使ってもらえるか。それが大事だと思う」

「みんな違う視点で生産性を見てるんですね...」

湊の呟きに飛鳥は頷いた。

「そうなんだよ。プロダクト部と開発部、QA部、それぞれが大事にしてるものが違う。それをどう調整するかが、私たちの仕事なんだけど」

飛鳥は少し間を置いて続けた。

「でも、それぞれの視点を理解することが第一歩だと思う。湊君がいろんな人に話を聞いてるって聞いたよ。それは良いことだと思う」

会議室を出た湊は、自分のデスクに戻りながら考えていた。

エンジニアは技術的品質を重視する。QAは品質保証を重視する。PMはユーザー価値とビジネス成果を重視する。

どれも正しい。でも、バラバラだ。どうやって統合すればいいのか。

湊はノートに書き出した。山田さんとの対話で学んだ「技術的負債の返済も生産性の一部」。高橋さんとのランチで学んだ「品質も生産性のうち」。飛鳥との議論で学んだ「ユーザー価値と持続可能性のバランス」。それぞれの視点は、それぞれの立場から見た「正しい」生産性の定義だ。でも、それらが統合されていない。だから問題が起きる。

湊はこの時、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ではないか。

田中部長との再面談

翌日の午後3時。再び湊は会議室に呼ばれた。前回とは違い、今回は湊の方から相談したいことがあった。

「湊君、前回の件を踏まえて、改めて生産性について話そう」

田中部長の表情は前回よりも柔らかかった。今回は湊の努力を評価しているようだった。

湊は緊張しながらも準備してきた質問を切り出した。

「部長の考える開発生産性って、具体的にはどういうものですか？」

田中部長は少し驚いた様子だったがすぐに答えた。

「経営層からは数字を求められるんだ。売上への貢献、コスト削減、市場投入速度。でも、現場の苦労もわかる」

「数字だけじゃ測れないものもありますよね？」

湊の質問に田中部長は深くため息をついた。

「そうなんだよ。技術的負債の返済とか、チームの健全性とか、数字にしにくいものが多い。でも、経営会議では『具体的な数値』を求められる」

田中部長は資料を開いた。経営会議で使われた資料だ。

「先月の経営会議で、開発部の生産性について質問された。『他社と比べてどうなのか』『数字で示せるのか』。でも、単純な数字では測れないものがある。それをどう説明すればいいのか、僕も悩んでるんだ」

「部長自身は何が大事だと思いますか？」

「...正直、バランスだと思う。短期的な成果と長期的な健全性。スピードと品質。でも、それをどう測るかが難しいんだ」

田中部長は窓の外を見た。

「例えば、技術的負債の返済。これは長期的な健全性のために必要だ。でも、その時間を確保するのは難しい。新機能の開発が優先されるから。結果として、技術的負債は蓄積し続ける。そして開発速度はさらに低下する」

「それでも、経営層には数字を求められる...」

「そうだ。だから難しい。短期的な数字を出すために、長期的な健全性を犠牲にする。でも、それでは持続できない。バランスが大事なんだ」

田中部長は湊を見た。

「湊君、いろんな人に話を聞いてるんだろう？それは良いことだ。リーダーとしていろんな視点を理解するのは大事だからね」

「はい。山田さん、高橋さん、飛鳥さんにも話を聞きました」

「どうだった？」

湊は少し考えてから答えた。

「山田さんからは、技術的負債の返済も生産性の一部だということを学びました。過去のデータを見ると、技術的負債の蓄積で開発速度が年間20-40%低下している。でも、それをどう上司に説明すればいいのか、山田さんも悩んでいると言っていました」

田中部長は深く頷いた。

「高橋さんからは、品質も生産性のうちだということを学びました。バグ修正コストは段階的に増大する。要件定義段階なら1倍、実装段階なら10倍、リリース後なら30-100倍。リリース後にバグが見つかると、チーム全員で徹夜対応することになる。でも、開発中にテストで見つけていれば、1人が数時間で修正できたはずだ。長期的に見ると、品質を重視した方が生産性が高い」

「それは重要な視点だね」

「飛鳥さんからは、ユーザー価値と持続可能性のバランスが重要だということを学びました。PMとして、機能リリース数は重要だ。でも、それだけじゃない。ユーザーが本当に価値を感じてくれるか。長期的に使ってもらえるか。それが大事だと言っていました」

田中部長は少し間を置いてから言った。

「みんなそれぞれ違う視点で生産性を見ている。エンジニアは技術的品質を、QAは品質保証を、PMはユーザー価値を重視する。でも、どれも正しい。そうなんだよ。だから難しい。共通の理解を作るのがリーダーの仕事なんだ。それぞれの

視点を理解して、統合する。それができれば、チーム全体の生産性が上がるはずだ」

湊はこの時、初めて理解した。リーダーとしての自分の役割は、各職種の視点を理解し、相乗効果を作ることなのだと。

田中部長は少し間を置いて、資料をめくった。

「実は、開発生産性を考える上で、もう一つ重要な視点があるんだ。それは『開発業務の最適化』だ」

「開発業務の最適化...ですか」

「うん。開発生産性を上げるには、開発そのものだけじゃなく、開発に関わる全ての業務を見直す必要がある。例えば、開発中のイニシャルコストを削減する。同じ機能を開発するのに、以前は10日かかっていたものを8日でできるようにする。これが継続的にできれば、長期的なコスト削減になる」

湊はメモを取りながら聞いていた。

「それから、ミーティングも見直せる。ムダな会議はないか。本当に必要な会議だけに絞る。会議時間を短縮する。これだけでも、開発に使える時間が増える」

「確かに会議が多いと感じることもあります」

「そうだろ。それから、保守運用の作業も見直せる。ムダな作業はないか。自動化できるものはないか。手作業でやっていることを、スクリプトで自動化する。これも開発生産性の向上につながる」

田中部長は資料のページをめくった。

「これらを金額に落とし込んでみると、結構な額になる。でも、無理に数字で測ろうとしすぎると、本質を見失うこともあるんだ」

「本質...ですか」

「うん。開発生産性は、結局のところ『信頼を作る』ことに価値があると思うんだ。チームメンバー同士の信頼、他部署との信頼、ユーザーとの信頼。この信頼関係が構築されれば、長期的な価値につながる。でも、これを数字で測るのは難しい」

湊は深く頷いた。確かに信頼関係は数値化しにくい。

「実は、事業責任者や経営層の視点で考えると、もっとシンプルなんだ。彼らが意図したタイミングで、意図した成果物が出てくれば、信頼が積まれる。そうすれば、開発生産性をチーム外から求めることはなくなるかもしれない」

田中部長は少し間を置いて続けた。

「逆に、常にプロジェクトが計画から遅延する。そうすると、一言目に『開発生産性はどうなってるんだ』という言葉が出てくる。信頼が失われているから、数字で測ろうとするんだ」

湊はその言葉を聞いて、先日の失敗を思い出していた。計画通りにリリースできなかった。その結果、田中部長から「開発生産性を向上させてほしい」と言われた。

「例えば、先月のプロジェクト。計画では3週間でリリースする予定だった。でも、実際には5週間かかった。その時、経営会議で『開発生産性はどうなってるんだ』という質問が出た。計画通りに進まないから、数字で測ろうとするんだ」

田中部長は資料をめくった。

「でも、計画通りに進んでいるプロジェクトでは、そんな質問は出ない。事業責任者も経営層も、『このチームは信頼できる』と思っている。だから、開発生産性をチーム外から求めることはない。結局は、意図したタイミングで意図した成果物が出てくれば、信頼が積まれるんだ」

田中部長は少し間を置いて、湊を見た。

「ただ、現実はそう簡単じゃない。何をなぜ作るかを考える事業責任者やPdMの要求や仮説は、一定の間隔で現れるわけじゃない。ボラタリティー、つまり変動するんだ。市場の状況が変われば、優先順位も変わる。ユーザーの反応を見て、方向性を変えることもある。だから、計画通りに進まないこともある」

湊は深く頷いた。確かに先日のプロジェクトでも、途中で仕様が変更された。市場の競合他社が似た機能をリリースしたため、急遽方向性を変えることになった。

「でも、だからこそ、信頼が重要になる。要求や仮説が変動しても、それに対応できる。意図したタイミングで、意図した成果物を出せる。それが信頼につながるんだ」

「だから、『儲かる』を意識することは大事だ。開発生産性の向上が、ビジネスにどう貢献するのか。でも、無理に数字で測ろうとしない。信頼関係の構築が、結果として長期的な価値につながる。これが僕の考えだ」

田中部長は湊を見た。

「湊君、数字を出すことも大事だ。でも、それだけじゃない。信頼を作ることも大事なんだ。そのバランスを取ることが、リーダーの仕事だと思うよ」

信頼という言葉の重み

会議室を出た湊は新たな課題を感じていた。それぞれの視点を理解することはできた。開発業務の最適化という視点も得られた。でも、どうやって統合すればいいのか。共通の理解を作るには、何が必要なのか。そして、信頼を作ることをどう実現すればいいのか。

解説：ステークホルダーごとに異なる「生産性」の定義

ストーリーで描かれる「重圧」

多角的な理解を求められる段階でよく現れるのが、「開発生産性の目的を求められる」という重圧です。湊が直面したのも同じ構造からです。

- **深夜の問い:**　「開発生産性って結局何なんだろう」。コード行数・リリース数・DORAメトリクスなど調べても、どれも一面的で答えが定まらない
- **職種ごとの定義の違い:** 山田（技術的負債・持続可能性）、高橋（品質・テスト時間）、飛鳥（ユーザー価値・市場対応）、田中部長（売上・コスト）。同じ「生産性」という言葉で、中身が違う
- **指標のズレ:** 物的生産性は数値化しやすいが品質を測れず、付加価値生産性は理想だが測定が難しい。「数字を出せ」と言われても、何を測ればよいか合意がない
- **コミュニケーションロス:** それぞれが「正しい」と言いながら噛み合わず、共通理解ができない

この重圧の背景には、**「生産性」の定義がステークホルダーごとにバラバラで、統合されていない構造**があります。まず「誰が何を生産性だと思っているか」を認識する段階でつまずいているのです。

なぜ視点によって定義が異なるのか

湊が気づいた「視点の違い」は、多くのチームが経験する典型的な問題です。

開発生産性は、立場によって重視する側面が異なります。

- **エンジニア:** コードの品質、技術的負債の管理、開発速度の持続可能性
- **QA:** バグの少なさ、テストカバレッジ、品質基準の達成
- **PM/PdM:** ユーザー価値の提供、機能リリースのスピード、市場への対応力
- **管理職:** 売上貢献、コスト効率、チーム全体のパフォーマンス

それぞれの視点は「正しい」が、統合されていないことで問題が起きます。エンジニアが技術的負債の返済を優先したいと考えても、PMは新機能のリリースを優先したいと考える。QAがテスト時間を確保したいと考えても、管理職はコスト

削減を優先する。

物的生産性と付加価値生産性の違い

生産性を考える際、「物的生産性」と「付加価値生産性」の違いを押さえておきます。

物的生産性は、投入量に対する産出量の比率で測られます。具体的には次のような指標があります。

- 労働時間あたりのコード行数
- 1週間あたりの機能リリース数
- 1日あたりのコミット数

これらは数値化しやすく、測定も容易です。しかし無駄なコードを書けば数字は上がりますし、リファクタリングでコードを削減すれば数字は下がります。品質や価値は測れません。

一方、**付加価値生産性**は、ビジネス価値やユーザー価値を生み出す効率性を測ります。

- ユーザーが実際に使う機能の開発速度
- ビジネス目標達成への貢献度
- 長期的な価値創出の効率性

これは理想的な指標ですが、測定が難しいという課題があります。ユーザー価値は数値化しにくく、ビジネス価値もすぐには見えません。

開発生産性を考える際は、物的生産性と付加価値生産性の両方を意識し、バランスを取ることが重要です。

開発業務の最適化という視点

田中部長が指摘したように、開発生産性を向上させるには、開発そのものだけでなく、開発に関わる全ての業務を見直す必要があります。

開発中のイニシャルコスト削減

- 同じ機能を開発する時間を短縮する
- 再利用可能なコンポーネントを作る
- 開発プロセスを改善する

これが継続的にできれば、長期的なコスト削減につながります。

ムダな会議の削減

- 本当に必要な会議だけに絞る
- 会議時間を短縮する
- 非同期コミュニケーションを活用する

これだけでも、開発に使える時間が増えます。

保守運用のムダな作業の削減

- 手作業でやっていることを自動化する
- スクリプトで自動化できるものは自動化する
- 運用プロセスを見直す

これらを金額に落とし込んでみると、結構な額になります。ただし、無理に数字で測ろうとしすぎると、肝心の目的を見失うこともあります。

信頼を作ることに価値がある

開発生産性は、結局のところ「信頼を作る」ことに価値があります。チームメンバー同士の信頼、他部署との信頼、ユーザーとの信頼。この信頼関係が構築されれば、長期的な価値につながります。

事業責任者や経営層の視点で考えると、もっとシンプルです。彼らが意図したタイミングで、意図した成果物が出てくれば、信頼が積まれます。そうすれば、開発生産性をチーム外から求めることはなくなるかもしれません。

逆に、常にプロジェクトが計画から遅延する。そうすると、一言目に「開発生産性はどうなってるんだ」という言葉が出てきます。信頼が失われているから、数字で測ろうとするのです。

ただし、現実はそう簡単ではありません。何をなぜ作るかを考える事業責任者やPdMの要求や仮説は、一定の間隔で現れるわけではありません。ボラティリティ、つまり変動します。市場の状況が変われば、優先順位も変わります。ユーザーの反応を見て、方向性を変えることもあります。だから、計画通りに進まないこともあります。

しかしだからこそ、信頼が重要になります。要求や仮説が変動しても、それに対応できる。意図したタイミングで、意図した成果物を出せる。それが信頼につながるのです。

「儲かる」を意識することは大事です。開発生産性の向上が、ビジネスにどう貢献するのか。でも、無理に数字で測ろうとしない。信頼関係の構築が、結果として長期的な価値につながるのです。

共通理解を作るために

まずは対話から始めましょう。湊が行ったように、各職種の視点を理解することから始める。

- 定期的な1on1で、相手が何を重視しているかを聞く
- チーム全体で「生産性とは何か」を議論する時間を作る
- 多面的な指標を組み合わせて測定する
- 開発業務の最適化の視点を取り入れる
- 信頼関係の構築を意識する

詳細な手法については、章末のステークホルダー視点マッピングのワークシートを参照してください。

2-2 エンジニアとして他者の視点を想像する重要性

帰りの電車での振り返り

帰りの電車内。湊はスマートフォンのメモアプリに今日の発見を記録していた。

視点の整理と気づき

山田さん、高橋さん、飛鳥さん、田中部長。それぞれの視点を聞いて、湊は新たな気づきを得ていた。

それぞれの立場で生産性の定義が違う

- **エンジニア（山田さん）**：技術的品質、持続可能性、技術的負債の返済
- **QA（高橋さん）**：品質、バグの少なさ、テスト時間の確保
- **PM（飛鳥さん）**：ユーザー価値、市場対応速度、ビジネス成果
- **管理職（田中部長）**：売上貢献、コスト効率、でも現場の健全性も理解

湊は画面を見つめた。それぞれの視点は、それぞれの立場から見た「正しい」生産性の定義だ。でも、それらが統合されていない。だから問題が起きる。

「どれも正しいけど、バラバラだから問題が起きるんだ」

エンジニアは技術的負債の返済を優先したい。でも、PMは新機能のリリースを優先したい。QAはテスト時間を確保したい。でも、管理職はコスト削減を優先する。

それぞれが正しい。でも、統合されていない。だから衝突が起きる。

「じゃあ、どうやって共通の理解を作ればいいんだろう？」

湊はメモアプリに書き続けた。

共通理解を作るために必要なこと

- それぞれの視点を理解する
- 多面的な指標を設定する
- 対話の仕組みを作る
- 長期的な視点を持つ

でも、これだけでは不十分だ。もっと具体的な方法が必要なのではないか。

湊は田中部長の言葉を思い出していた。開発業務の最適化。開発中のイニシャルコスト削減、ムダな会議の削減、保守運用のムダな作業の削減。これらを金額に落とし込んでみると、結構な額になる。

でも、無理に数字で測ろうとしすぎると、本質を見失う。田中部長が言っていた「信頼を作る」ことの重要性。事業責任者や経営層が意図したタイミングで意図した成果物が出てくれば、信頼が積まれる。逆に、常に計画から遅延すれば、信頼が失われ、数字で測ろうとするようになる。

湊はメモアプリに追記した。

開発業務の最適化

- 開発中のイニシャルコスト削減
- ムダな会議の削減

- 保守運用のムダな作業の削減

信頼を作ることに価値がある

- 意図したタイミングで意図した成果物を出す
- 計画から遅延しないことが信頼につながる
- 無理に数字で測ろうとしない

電車が駅に到着し、湊は降りた。自宅への道を歩きながら新たな疑問が湧いてくる。

それぞれの視点を理解することはできた。開発業務の最適化という視点も得られた。信頼を作ることの重要性も理解した。でも、それをどう統合すればいいのか。エンジニアとして、他者の視点を想像することの重要性はわかった。でも、それだけでは不十分だ。もっと具体的な方法が必要だ。

解説：エンジニアとして他者の視点を想像する重要性

ストーリーで描かれる「重圧」

「視点は理解したが、どう統合すればいいかわからない」。一見すると個人の悩みのように見えますが、背景には**共通理解を構築する前の段階**に共通する構造的な課題があります。湊が経験した重圧もそこにつながっています。

- **視点は理解したが統合できない:** 山田・高橋・飛鳥・部長の「生産性」の違いは理解した。しかし「どれも正しいけど、バラバラだから問題が起きる」という状態で止まっている
- **共通理解の方法がわからない:** 「どうやって共通の理解を作ればいいんだろう？」という問いが残る。多面的な指標・対話・長期視点は必要だとわかっても、具体的手順が見えない
- **エンジニア視点の限界:** 自分（エンジニア）の価値観だけで進めると優先順位

の衝突・言葉のすれ違い・目標の不一致が起きる。他者の視点を「想像する」ことの重要性を痛感している

- **開発業務最適化と信頼の統合:** 部長の「開発業務の最適化」「信頼を作る」という視点は得たが、他職種の視点とどう統合し、次のアクションに落とすかがまだ見えていない

この重圧の背景には、**視点を「知る」ことと、それを「統合して共通理解にする」ことの間にギャップがある構造**があります。他者の視点を想像し、共通言語を作る段階に足を踏み入れようとしているのです。

なぜ他者の視点を理解するのか

湊が各職種にヒアリングして気づいたのは、「エンジニアとして他者の視点を想像することの重要性」です。

エンジニアだけの視点で開発を進めると、以下の問題が起きます。

- **優先順位の衝突:** エンジニアは技術的負債を返済したいが、PMは新機能を優先したい
- **コミュニケーションロス:** 同じ「生産性」という言葉を使っていても、意味が異なる
- **目標の不一致:** チーム全体で同じ方向を向けない

他者の視点を理解することで、それぞれの立場から見た「正しい」生産性の定義を理解できます。そして、それらを統合することで、共通の理解を作ることができます。

他者の視点を想像するために

他者の視点を想像するには、以下のことが重要です。

- **各職種との対話:** 湊が行ったように、各職種と1on1で話を聞く
- **相手の立場に立って考える:** なぜその視点が重要なのかを理解する

- **共通言語の確立:** 「生産性」という言葉の定義を明確にする

開発業務の最適化という視点

田中部長が指摘した開発業務の最適化（2-1 参照）も、他者の視点の一つとして押さえておく。信頼を作るという視点も 2-1 の「信頼を作ることに価値がある」を参照する。

次のステップ

湊が次に考えるのは、「持続可能な開発とは何か」です。それぞれの視点を理解しても、それらを統合する方法がわからない。持続可能な開発を実現するには、何が必要なのか。

次のセクションでは、この「持続可能な開発」について詳しく見ていきます。

2-3 持続可能な開発とは何かを模索する

自宅での静かな夜

自宅に戻り、湊はノートPCを開いた。

持続可能性への気づき

リビングのテーブルには空き缶が3本だけ。先日の失敗直後と比べると、少し落ち着いてきた証拠だ。窓際のパキラも水をやったばかりで、葉が少し元気を取り戻している。

湊は今日一日で得た気づきを振り返っていた。山田さん、高橋さん、飛鳥さん、田中部長。それぞれの視点を聞いて、湊は生産性という言葉が持つ多面性を理解した。

でも、それだけでは不十分だ。それぞれの視点を理解しても、統合する方法がわからない。共通の理解を作るには、何が必要なのか。

湊は過去のプロジェクトデータを振り返る。開発速度の推移、バグの発生率、残業時間。様々な数字が画面に並ぶ。

去年の同じ時期、チームの平均残業時間は20時間だった。でも、今年は40時間を超えている。開発速度は低下し、バグの発生率は増加している。

グラフを見ると、明確なトレンドが見える。開発速度は月ごとに1-2%ずつ低下している。バグの発生率は逆に上昇している。残業時間も増加の一途だ。

「これは持続可能な開発なのか？」

湊は自問した。短期的には機能をリリースできている。でも、長期的に見ると、チームは疲弊している。開発速度は低下している。バグの発生率は増加している。

このままでは、さらに悪化するのではないか。技術的負債は蓄積し続け、開発速度はさらに低下する。チームメンバーは疲弊し、離職する人も出てくるかもしれない。

これは持続可能な開発なのだろうか。

「測れない価値」について考え始める。

技術的負債の影響、品質向上の効果、チームの健全性。これらをどう可視化すれば、みんなが共通の理解を持てるのだろうか？

湊はノートに書き出し始めた。

持続可能な開発とは何か

- 短期的な成果だけでなく、長期的な健全性も考慮する
- 技術的負債を適切に管理する
- 品質を維持しながら開発速度を保つ

- チームの健全性を守る

　でも、これらをどう測ればいいのか。どう可視化すれば、みんなが共通の理解を持てるのか。

　湊は画面を見つめた。過去のプロジェクトデータには、様々な数字が並んでいる。でも、それだけでは不十分だ。もっと多面的な指標が必要なのではないか。

見えない生産性を見える化する方法

　技術的負債の影響、品質向上の効果、チームの健全性。これらをどう可視化すれば、みんなが共通の理解を持てるのだろうか？

　まだ答えは見えない。でも、問いが明確になった。

　持続可能な開発とは、短期的な成果だけでなく、長期的な健全性も考慮することだ。技術的負債を適切に管理し、品質を維持しながら開発速度を保ち、チームの健全性を守ることだ。

　でも、それをどう実現すればいいのか。どう可視化すれば、みんなが共通の理解を持てるのか。

　窓の外を見ると、夜空に星が見える。湊は深呼吸をした。

「明日、もう一度山田さんに相談してみよう。見えない価値を可視化する方法について」

解説：持続可能な開発とは何か

ストーリーで描かれる「重圧」

　前のシーンで湊が経験した「このままでは持続できない」という壁は、**持続可能性を問う段階**における構造的な課題から生じています。

- **データが示す持続不可能性:**　残業時間（20時間→40時間超）、開発速度の低下、バグ発生率の増加。グラフが「持続可能な開発ではない」ことを示している
- **技術的負債と体制の追いつかなさ:** 技術的負債の蓄積で開発速度が落ち、スケジュールだけが先行しチームの体制・スキルが追いつかない。このままでは悪化し、離職もあり得る
- **見えない価値の可視化の壁:** 技術的負債の影響・品質向上の効果・チームの健全性を「どう可視化すれば、みんなが共通の理解を持てるか」がわからない
- **短期と長期のバランス:** 短期的にはリリースできているが、長期的な健全性をどう測り・どう話すかが未確立で、持続可能な開発への道筋が描けない

この重圧の背景には、**短期的成果は出ている一方で、長期的健全性を測る指標と共通言語がなく、「持続可能か」を説明できない構造**があります。第3章「見えない価値の可視化」へつなぐ転換点に立っているのです。

なぜ持続可能性を押さえるのか

湊が最後に気づいた「持続可能性」は、開発生産性を考える上で押さえるべき視点の一つです。

短期的な成果だけを追求すると、以下の問題が起きます。

- **技術的負債の蓄積:** 急いで開発した結果、コードの品質が低下し、将来的な開発速度が低下する
- **チームの疲弊:** 残業が続き、メンバーのモチベーションが低下する
- **品質の低下:** テスト時間を削減した結果、バグが増加し、後で修正コストがかかる

持続可能な開発とは、短期的な成果だけでなく、長期的な健全性も考慮することです。

持続可能な開発に必要なこと

持続可能な開発を実現するには、以下の要素が必要です。

- **短期的な成果と長期的な健全性のバランス:** 新機能の開発と技術的負債の返済を両立する
- **技術的負債の適切な管理:** 定期的にリファクタリングの時間を確保する
- **チームの健全性の維持:** 残業時間を適切に管理し、メンバーのモチベーションを維持する
- **品質の維持:** テスト時間を確保し、バグの発生を防ぐ

次のステップ

湊が次に取り組むのは、「見えない価値の可視化」です。技術的負債の影響、品質向上の効果、チームの健全性をどう測定し、共通の理解につなげるのか。

次の章では、この「見えない生産性を見える化する」方法について詳しく見ていきます。

手法2 ステークホルダー視点マッピング

湊が行ったように、各職種の視点を理解することは、共通理解を作る第一歩です。このセクションでは、具体的な手法を紹介します。

ステークホルダーマップの作成

目的: チームに関わる各職種が何を重視しているかを可視化する

手順:

1. 関係者をリストアップする（エンジニア、QA、PM、管理職、営業、サポートなど）

2. 各職種が「生産性」に関して何を重視しているかをヒアリング
3. 表やマップにまとめる

テンプレート例:

職種	重視する指標	測定方法	懸念事項
エンジニア	コード品質、技術的負債	コードレビュー、静的解析	技術的負債の増加
QA	バグの少なさ、テストカバレッジ	バグ数、カバレッジ%	テスト時間の不足
PM	ユーザー価値、リリース速度	機能リリース数、ユーザー満足度	市場競争力の低下
管理職	売上貢献、コスト効率	売上、工数	予算超過

対話のテンプレート

各職種と対話する際の質問例です。

エンジニアへの質問:

- 現在の開発で一番の課題は何ですか？
- 技術的負債が開発速度にどう影響していますか？
- 理想的な開発環境とはどういうものですか？

QAへの質問:

- テスト工数は十分に確保されていますか？
- バグが多い原因はどこにあると思いますか？
- 品質を保つために必要なことは何ですか？

PMへの質問:

- ユーザーにとって最も価値のある機能は何ですか？
- リリースのスピードと品質、どちらを優先すべきですか？
- ビジネス目標を達成するために必要なことは何ですか？

管理職への質問:

- 経営層から求められている目標は何ですか？
- 現場の課題をどう理解していますか？
- 短期的な成果と長期的な健全性、どうバランスを取りますか？

実践のステップ

ステップ1: 1on1の実施

- 各職種と30分〜1時間の1on1を実施
- 上記の質問を使ってヒアリング
- メモを取り、後でまとめる

ステップ2: マップの作成

- ヒアリング内容を表やマップにまとめる
- 視点の違いを明確にする
- 共通点と相違点を可視化する

ステップ3: チーム共有

- チームミーティングで共有
- 「それぞれの視点を理解する」ことを目的にする
- 批判ではなく、理解と対話を促す

ステップ4: 共通理解の構築

- 各職種の視点を統合した「共通の定義」を作る
- 多面的な指標を設定する
- 定期的に見直す（3ヶ月ごとなど）

チェックリスト

実施前:

- [ ] 関係者をリストアップした
- [ ] 1on1の時間を確保した
- [ ] 質問を準備した

実施中:

- [ ] 各職種の視点を理解できた
- [ ] メモを取り、整理した
- [ ] 共通点と相違点を見つけた

実施後:

- [ ] マップを作成した
- [ ] チームに共有した
- [ ] 共通の定義を作り始めた

第2章のまとめ

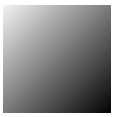
湊は、各職種へのヒアリングを通じて、「開発生産性」という言葉が持つ多面性に気づきました。

主な学び:

- エンジニア、QA、PM、管理職、それぞれ異なる視点で生産性を見ている
- どの視点も「正しい」が、統合されていないことで問題が起きる
- エンジニアとして他者の視点を想像することの重要性
- 持続可能な開発とは、短期的な成果だけでなく、長期的な健全性も考慮すること

次章への課題:

- 「見えない価値」をどう可視化するか
- 技術的負債、品質向上の効果、チームの健全性をどう測定するか
- 持続可能な開発を実現するための具体的な方法



次の章では、湊が「測れない生産性」にどう向き合うのかを見ていきます。