

# ● 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

午後10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。しかし湊は一人残り、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。先日のチームミーティングでは開発生産性について話し合ったが、定義はまだ漠然としたままだ。

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。

「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。「開発生産性を可視化し、向上させてほしい」。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。答えが定まらないまま、湊はブラウザを開き、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、湊は首を横に振る。コード行数だけでは、無駄なコードを書けば数字は上がり、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…」

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。コード行数、リリース数、ストーリーポイント——どれも数値は出せるが、品質や価値は測れない。

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。

湊はコーヒーを淹れに行きデスクに戻った。オフィスには誰もおらず、静まり返った空間で、湊は改めて考える。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が、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。

## エレベーターでの偶然の出会い

翌朝8時、湊は早めに出社した。答えは見つからなかったので、山田に相談しよう決めていた。エレベーターホールで、彼と偶然一緒になった。朝の光がガラス越しに差し込み、まだ人の気配が少ない時間帯だ。

「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つた。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で、自動販売機のコーヒーを手に立ち話が始まる。

「山田さん、実は昨日、開發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...」

湊の質問に山田は腕を組み、少し考えてから答える。

「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かなり低下してる。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、今は明らかに時間がかかってる。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。去年の同じ時期は数日で済んでいた機能開発が、今年は同じ規模にもっとかかっており、開発時間が伸びているのだ。

「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み、少し声を落とした。

「正直さ、僕は技術の話なら何時間でもできるけど、経営層に伝わる言葉で話すのが苦手なんだ。前の会社でも、リファクタリングの必要性を説明しようとして、結局『で、売上にどう効くの？』と返されて黙ってしまった。それ以来ずっ

と、この壁を超えられていない」

山田は少し間を置いてから、いつもの調子で話を続ける。

「例えば、先月のプロジェクト管理モジュールの改修。本来なら数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。想定よりずっとかかった」

「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。「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。話し声と食器の音が響くなか、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る。

「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、体感10倍はかかるんですよ。バグ修正コストは段階的に増大するんです」

「段階的に？」

「感覚ですが、段階が遅くなるほど、修正コストは何倍にも跳ね上がるんです。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くんです」

高橋は箸を手に取り、ご飯を一口食べる。

「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。

## カスタマーサポートの鈴木との廊下での立ち話

午後3時。湊が自販機にコーヒーを買いに行くと、カスタマーサポート部の鈴木一郎と鉢合わせた。

「あ、鈴木さん。お疲れ様です」

「湊さん、ちょうどよかった。聞きたいことがあったんだ」

鈴木は缶コーヒーのプルタブを開けながら、淡々と話し始めた。

「先週のリリース後、問い合わせが3倍に増えた。うちのチーム、3人で回してるんだけど、全員が残業してる。ユーザーは『前まで使えてた機能が動かない』って怒ってる。新機能よりも、今使えてるものが壊れない方がよっぽど大事なんだよ、ユーザーにとっては」

湊は言葉に詰まった。開発チームの目線ではリリースできたことが「成果」だった。でも、サポートの現場ではそれが「問題の始まり」だった。

「生産性って、サポートの視点だとどう見えますか？」

「うーん、『ユーザーが困らないこと』かな。機能がいくら増えても、問い合わせが増えたら僕らの仕事は増える。品質が高ければ問い合わせは減る。それが一番の生産性だと思うよ」

湊は頷いた。山田の技術的負債、高橋の品質、そして鈴木ของผู้ユーザー視点。それぞれの立場で「生産性」の意味が違う。その実感が、また一つ増えた。

## 飛鳥との夕方の対話

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。窓の外はまだ明るいけど、室内の照明だけがテーブルを照らしている。

「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。昨日のような対応の末に、これが「良かった」と言えるのか。口にした疑問が、そのまま胸に残る。



「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。しばらく窓の外を見てから、いつもと違う静かな声で言った。

「……正直に言うとな、この前のリリースの後、サポートチームの鈴木さんに『もう少し品質を上げてから出してほしい』って言われたんだ。ユーザーからのクレーム対応で、鈴木さんのチームも疲弊していた。私がスピードを優先した結果、他のチームにまで迷惑をかけていた。それがすごくこたえた」

飛鳥は少し息をついてから、同じ静かな声で続けた。

「……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す。

「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み、例を挙げる。

「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る。

「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く。

「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」

飛鳥は少し間を置き、同じ調子で続ける。

「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、廊下を歩きながら自分のデスクに戻る。冷房の音が響く廊下で、頭のなかを言葉が駆け巡る。技術的負債、品質、ユーザー価値——どれも正しいのに、バラバラで噛み合わない。生産性という言葉は立場によって意味が変わる。

## 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。ドアをノックして中に入ると、田中部長が窓際の席で待っている。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、すぐに答える。

「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く。しばらく黙ってから口を開いた。

「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」

田中部長は経営会議で使われた資料を開いた。

「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。

「技術的負債の返済やチームの健全性は、必要だとわかっていても数字にしにくい。経営会議では説明が難しいんだ」

「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？」

湊は三人の話の要点を短くまとめた。山田さんは技術的負債の返済も生産性の一部だと。高橋さんは品質を重視した方が長期的には生産性が高いと。飛鳥さんはユーザー価値と持続可能性のバランスが大事だと。

田中部長は少し間を置いてから口を開いた。

「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、開發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『開業務の最適化』だ」

「開業務の最適化...ですか」

「うん。開發生産性を上げるには、開そのものだけでなく、開に関わる全ての業務を見直す必要がある。例えば、開中のイニシャルコストを削減する。同じ機能を開するのに、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。これも開發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むと、無視できない額になる。ただ、数字だけ追うと、肝心の目的を見失う。開發生産性でいちばん大事なのは、信頼を作ることだと思うんだ」

「信頼、ですか」

「うん。開發生産性は、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」

湊は深く頷く。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き、同じ調子で続ける。

「逆に、計画から遅延が続くと、『開発生産性はどうなってるんだ』と聞かれる。信頼が失われているから、数字で測ろうとする。先月も、3週間の予定が5週間かかって、経営会議でそう言われた。逆に、意図したタイミングで成果物が出れば、そんな質問は出ない。結局、信頼が積まれるかどうかが大事なんだ」

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。

「ただ、現実はその簡単じゃない。事業責任者やPdMの要求は変動する。市場やユーザーの反応で優先順位が変わる。計画通りに進まないこともある。だからこそ、要求が変わっても対応できることが信頼につながる」

湊は深く頷く。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。

「『儲かる』を意識しつつ、無理に数字で測りすぎない。信頼を作ることが、長期的な価値につながる。これが僕の考えだ」

田中部長は少し間を置いてから湊を見た。

「湊君、数字を出すことも大事だ。でも、それだけじゃない。信頼を作ることでも大事なんだ。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、廊下に足を踏み出し、何歩か歩いた頃に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、信頼を作ることとをどう実現すればいいのか。部長の「信頼を作る」という言葉が、胸のなかで重く残っている。

---

## 解説：ステークホルダーごとに異なる「生産性」の定義

この節では、生産性の定義が職種ごとに異なること、物的生産性と付加価値生産性の違い、そして信頼を作るという視点を整理します。

### ストーリーで描かれる「重圧」

山田は技術的負債を、高橋は品質を、飛鳥はユーザー価値を、部長は売上を語った。同じ「生産性」という言葉で。

- **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：** 立場ごとに「正しい」と言いながら、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づいているのです。

### 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ます。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します。

生産性を多角的に捉えるうえでは、研究で使われている枠組みが参考になります。DORA メトリクスでは、デプロイ頻度・変更リードタイム（スピード）と、変更失敗率・復旧時間（安定性）を**対**で捉え、どちらか一方だけを追うのではなく両方のバランスを見ることを推奨しています。また SPACE という枠組みでは、開発者生産性を満足度（Satisfaction）・パフォーマンス（Performance）・活動量（Activity）・コミュニケーション（Communication）・効率（Efficiency）の多次元で扱うべきだとしています。一つの数値だけでなく、複数の軸を組み合わせることで、湊が感じた「どれも一面的だ」という違和感を解きほぐすヒントになります。

## 欠陥修正コストの段階的増大（エビデンス）

ストーリーで高橋が「段階が遅くなるほど修正コストは何倍にも跳ね上がる」と感覚で話していた部分は、研究でも裏付けられています。

米国標準技術研究所（NIST）のレポート *The Economic Impacts of Inadequate Infrastructure for Software Testing*（2002年）では、欠陥を発見・修正する相対コストがライフサイクルの段階ごとに増大することが示されています。IBM Rational のホワイトペーパー *Minimizing code defects to improve*



*software quality and lower development costs* (2008年) の Figure 1 は、この NIST の知見を引用し、次のような相対コスト（基準を 1X として）を掲げています。

- 設計・アーキテクチャ: 1X (基準)
- 実装: 5X
- 結合テスト: 10X
- 顧客ベータテスト: 15X
- リリース後: 30X

つまり、リリース後に見つかった欠陥の修正コストは、設計段階で見つけた場合の最大で約30倍になるという試算です。ストーリーでの「体感10倍」「段階が遅いほど何倍にもなる」という感覚は、こうした研究結果と整合的です。

## 技術的負債と開発速度（エビデンス）

ストーリーで山田が「開発速度がかなり低下している」と感覚で語っていた部分も、技術的負債が開発に与える影響についての研究と整合的です。

複数の研究では、技術的負債が開発速度やリードタイムに与える影響が数値で報告されています。例えば、開発者の時間の 23～42% が技術的負債関連の活動に費やされているとする研究 (*Software developer productivity loss due to technical debt—A replication and extension study* 等) や、低品質なコードベースでは高品質なコードに比べて開発に要する時間が約2倍（124%増）になるという分析があります。また、産業ケーススタディでは、技術的負債がリードタイムのばらつきの 5～41% を説明するという結果も報告されています。

「技術的負債が蓄積すると開発速度が落ちる」という現場の感覚は、こうしたエビデンスと矛盾しません。定量的な根拠が必要な場面では、上記のようなレポート名と数値を参照して説明することが有効です。

## 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

田中部長が語ったように、開發生産性でいちばん大事なのは「信頼を作る」ことです。第1章で触れた約束の信頼性（計画したタイミングで成果物を出し、遅延が続くと信頼が失われる）と重なります。意図したタイミングで成果物を出し、要求が変動しても対応できることが信頼につながる——ストーリーで部長が話した内容を、共通理解を作るための対話や指標の選び方とあわせて、章末のステークホルダー視点マッピングでも扱います。

### 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ます。

- 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。揺れる車両のなか、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、立場から見た「正しい」定義だ。だが、統合されていないから噛み合わない。「どれも正しいのに、バラバラで問題が起きるんだ」と呟き、メモアプリに指を動かす。

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。

「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。

部長の「信頼を作る」「開発業務の最適化」という言葉が頭に残る。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。夜の駅前の明かりを背に、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が、それだけでは不十分で、もっと具体的な方法が必要だと感じている。

---

## 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いても、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」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る、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」定義を理解し、統合して共通理解を作る手がかりが得られます。

研究では、自己評価の生産性と強く相関するのはツールや環境よりも、仕事への**熱意・ピアサポート・有益なフィードバック**といった非技術的要因であるという結果があります。仕事満足度と知覚的生产性には双方向の因果があり、他職種との対話や互いの支えが、生産性の実感や持続可能な開発につながるという示唆です。湊が山田・高橋・飛鳥・部長にヒアリングして視点を広げたことは、こうした「他者とのつながり」を強める第一歩になっています。

## 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、湊はリビングのテーブルにノートPCを開いた。今日、飛鳥さんと部長の話のなかに出てきた「持続可能性」という言葉が、頭に残っていた。

## 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ず、共通の理解を作るには何が必要なのか、まだ手がかりが掴めない。

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。出力量だけでなく、集中して取り組める時間や状態が持続可能性には効く——その指摘が、ふと頭に浮かんだ。

去年の同じ時期に比べて、チームの残業はかなり増えている。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。開発速度は少しずつ落ち、バグの発生率は上がり、残業時間も増加の一途だ。

「これは持続可能な開発なのか？」

湊は自問する。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、開発速度は低下し、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。測り方がまだ見えない。

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方法

まだ答えは見えない。それでも、問いははっきりしてきた。ノートに書いた考えを、改めて心で繰り返す。

でも、それをどう実現すればいいのか。まだ答えは見えない。

窓の外を見ると、夜空に星が見える。湊は深呼吸をし、画面のグラフを一度閉じた。

「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」

見えない価値を可視化することは、共通理解や説明可能性を高める第一歩だ。その意志を胸に、湊は画面を閉じた。

翌朝、出社するとSlackに飛鳥からのメッセージが届いていた。

飛鳥さくら が #project-dev でメッセージを送信

湊君、昨日の話を考えたんだけど、次のプロジェクトでは開発に入る前に仕様レビュー会を設けない？  
先に認識を合わせてから進めたいんだ。

湊は画面を見つめた。昨日、飛鳥さんに話を聞きに行ったことが、こんなに早く形になるとは思っていなかった。対話に意味がある。その手応えが、胸の奥で小さく灯った。

---

## 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」

残業時間のグラフは右肩上がり、開発速度のグラフは右肩下がり。二つの線が交差する先に何があるのか。

- **データが示す持続不可能性：** 残業時間の増加、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。

なお、残業の増加や開発速度の低下といった傾向は、技術的負債や品質圧力の影響について述べた研究（本節 2-1 の解説「技術的負債と開発速度（エビデンス）」を参照）とも整合的です。

### 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、開發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。

開発者体験（DevEx）の研究では、生産性を「アウトプットの量」だけで測るのではなく、**フィードバックループ・認知負荷・フロー状態**の三つの軸で捉えることが提唱されています。とくに、アウトプット量よりも「フロー状態を維持できる時間」の長さに、生産性の本質が現れるとされています。湊が「残業が増え、開発速度は落ち、バグは増えている」とデータに気づいたように、持続可能な開発には、出力量だけでなく、チームが集中して取り組める状態をどう維持するかという視点が欠かせません。

## 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:** 定期的な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

帰りの電車で湊がメモアプリに書いた視点の整理は、ステークホルダー視点マッピングの最初の一步でした。以下のワークシートは、湊のように各職種の視点を体系的に整理するためのものです。

### 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（可視化は、共通理解や説明可能性を高める第一歩になる）
- 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。

## 参考文献

- DORA (DevOps Research and Assessment) . DORA metrics guide. <https://dora.dev/guides/dora-metrics/>（デプロイ頻度・変更リードタイムと変更失敗率・復旧時間を対で捉える指標）。
- Forsgren, N. et al. "The SPACE of Developer Productivity." ACM Queue, 2021. 開発者生産性を Satisfaction / Performance / Activity / Communication / Efficiency の多次元で扱う枠組み。
- merged\_references §5 開発者体験・ウェルビーイング：仕事満足度と知覚的生産性の双方向の因果、自己評価の生産性と相関する仕事への熱意・ピアサポート・有益なフィードバック。
- DevEx (Developer Experience) ：フィードバックループ・認知負荷・フロー状態の3次元、アウトプット量より「フロー状態を維持できる時間」を重視する考え方（merged\_references §2 に要約）。

