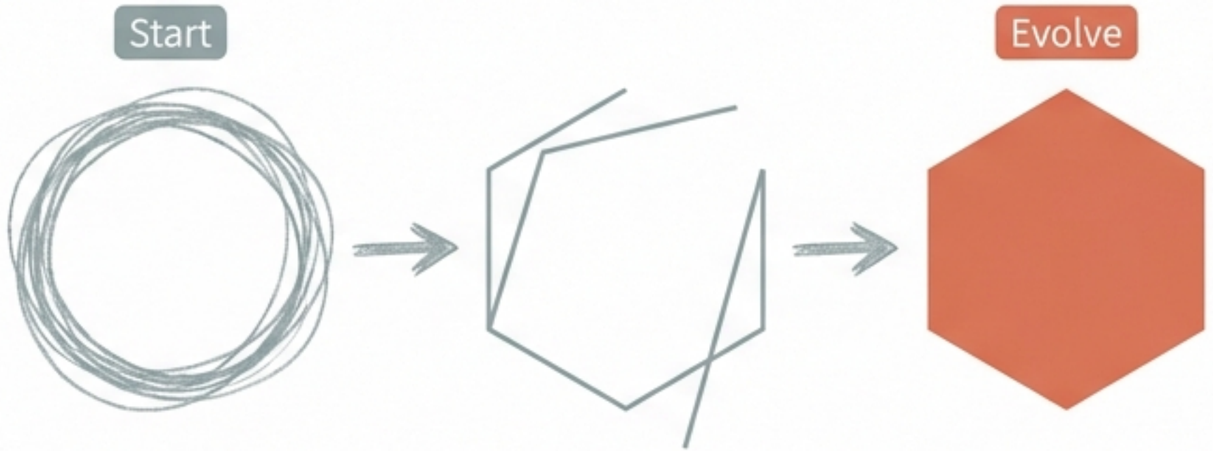
80点でリリースする勇気

完璧主義からの脱却

なぜ「まず動かす」が現代において最強の戦略なのか

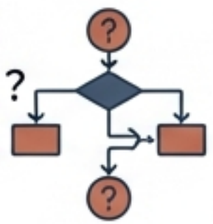


「すべてのパターンに対応したい」「例外も自動化したい」という思いは、プロフェッショナルとしての誠実さの表れです。しかし、その真面目さがプロジェクトを停滞させる「完璧主義の罠」となることがあります。本資料では、技術スキルではなく、成果を出し続けるための「マインドセットの転換」について解説します。

あなたを縛り付ける「ちゃんとしたものを作りたい」という呪縛

症状1：「あのパターンも！」症候群

「もしあのパターンが来たらどうする？」「例外が発生したら？」と、開発前から全ての可能性を網羅しようとしてしまう。



症状2：例外への過度なこだわり

実務で発生する稀な「例外」までも律儀にシステムでカバーしようとし、いつまでも仕様が固まらない。

症状3：リリースへの抵抗感

「不完全なものを出すのは恥ずかしい」「プロとして失格ではないか」という心理的ハードルが、完了を阻む。

Key Insight: これらは怠慢ではなく、「真面目さ」から来るブレーキです。しかし、現代のアジャイルな環境では、このブレーキがプロジェクトのリスクとなります。

完璧主義が引き起こす「見えないコスト」と「先延ばしの方程式」

プロジェクトへの悪影響

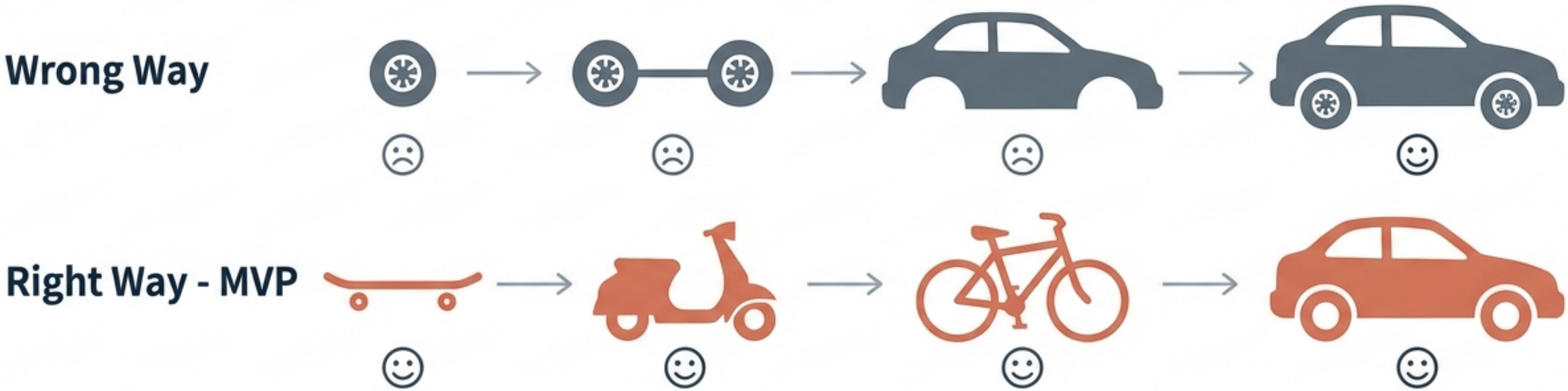
- 開発期間が長期化し、その間にビジネス環境や要件が変化してしまう。
- 「終わりの見えないマラソン」により、モチベーションが枯渇する。
- 最悪の場合、プロジェクト自体が頓挫（中止）する。

心理学的メカニズム

- タスクが難しすぎる・時間がかかりすぎる・失敗への恐怖恐怖がある。この３つが揃うと、人は無意識に「先延ばし」を選択します。
- 完璧を目指すほどタスクは巨大化し、着手すらできなくなります。

結論: 100点を目指すことは、結果として「0点（未完了）」のリスクを最大化させます。

MVP（Minimum Viable Product）の正しい理解



定義: 価値を検証するために必要な、最小限の機能を持った製品。
「これ以上削ると価値がなくなる」という最小単位です。

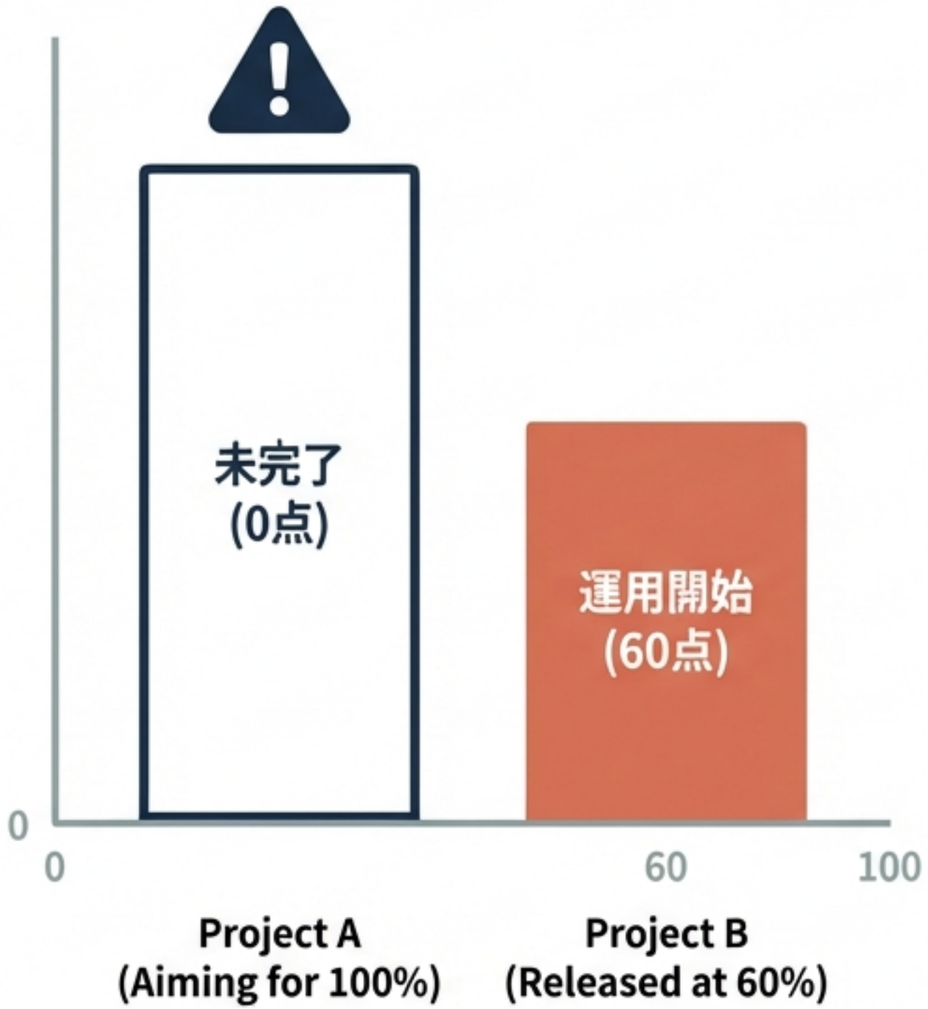
早期に価値を届ける:
完璧を待たずに、まずユーザーに使ってもらう。

フィードバックを得る:
机上の空論ではなく、実際の利用から本当に必要な機能を学ぶ。

リスクを分散する:
大きく作って大失敗するのではなく、小さく作って軌道修正する。

Japanese Modern Editorial

「80点でも合格」とする数学的根拠



現実1：メインケースで勝負が決まる

業務の80%（メインケース）を自動化・効率化できれば、残りの20%が手動であっても、全体としては劇的な改善になります。

現実2：やってみないと正解はわからない

「必須だ」と思っていた機能が実は不要だったり、想定外のニーズが見つかったりします。リリース前の推測は往々にして外れます。

価値の算数

完了しないプロジェクトの価値 = 0点
60点でリリースし、運用を開始したプロジェクトの価値 = 60点

結論: 完璧を目指して完了しないより、不完全でも完了させた方が、はるかに価値があります。

「段階的リリース」という戦略的アプローチ

Phase 1: コア機能

カバー率 50-60%

最も頻度の高いケースのみ対応。
「これだけでも助かる」状態。

Phase 2: パターン拡張

カバー率 70-80%

次に多いパターンを追加。
Phase 1のフィードバックを反映。

Phase 3: 例外対応

必要に応じて

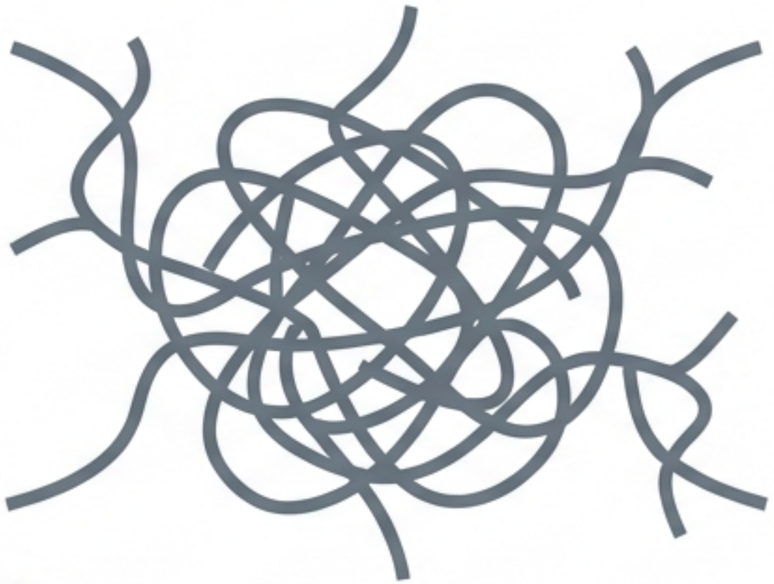
本当に必要か判断してから実装。
場合によっては「システム化しない」も正解。

メリット: ユーザーは早期に効果を実感でき、開発側は手戻りのリスクを最小化できます。

Japanese Modern Editorial

木を見て森も見る：拡張性を考慮した設計

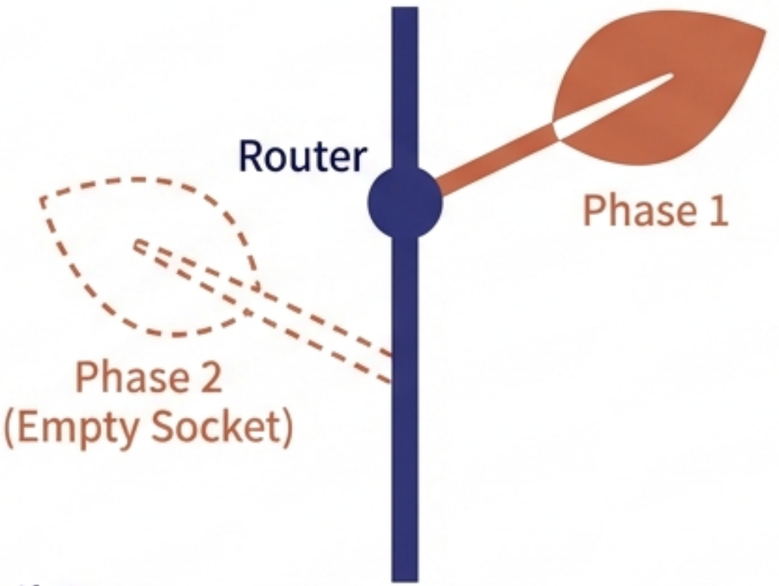
悪い例（Bad Example）



Noto Serif JP

条件分岐を一切考慮せず、ベタ書きで作ってしまう。
→ 修正が困難。

良い例（Good Example）



Noto Serif JP

将来の拡張を見越して「分岐点（Router）」だけは
最初から配置し、ルートは1つだけ実装する。

Noto Serif JP

解決策: 最終段階のあるべき姿（森）を想像しつつ、最初のゴール（木）を作ります。
フェーズ1のシステムが、フェーズ2・3でそのまま「拡張」できる土台を作っておくことが重要です。

実践ケーススタディ：問い合わせ自動化のスコープ

Phase 1 (MVP) - DO NOW

- [x] 通常対応（スプレッドシート記録）

  → 「問い合わせデータが漏れない・消えない」という最低限かつ最大の価値を担保。

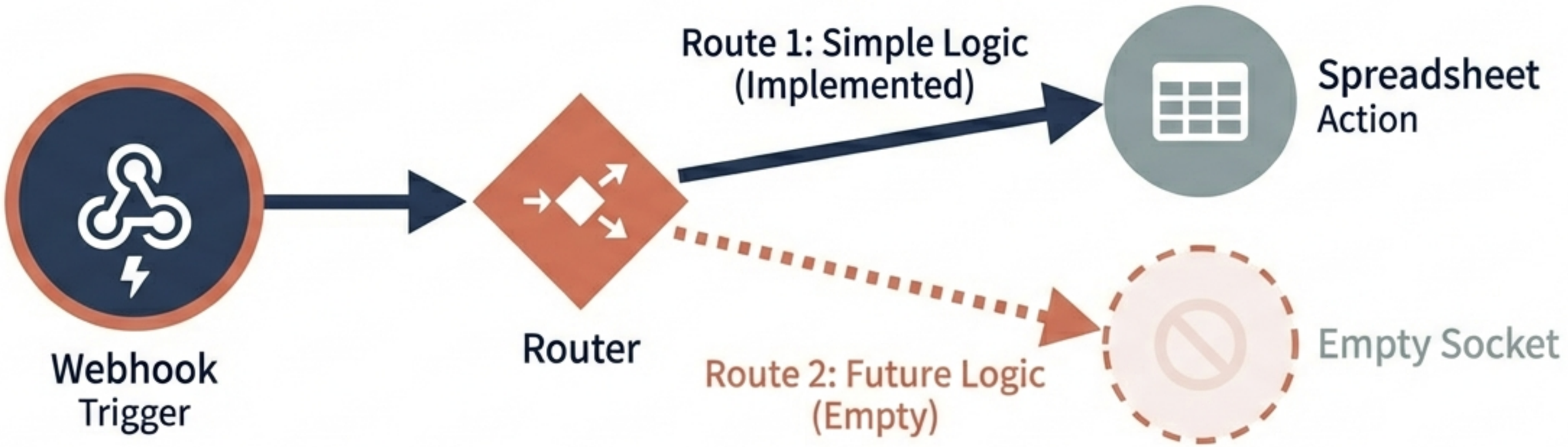
Phase 2 & 3 - LATER

- [ ] Slackへの緊急通知（Speed）
- [ ] 自動返信メール（Satisfaction）
- [ ] 担当振り分け（Efficiency）
- [ ] レポート作成

ポイント:

最初から複雑なAIボットや振り分けロジックを作ろうとしないこと。
理想の全要件を書き出した後、勇気を持って「今はやらない」を決断します。

Make.com等での「拡張しやすい」実装テクニック



1. **最初からRouterを配置する**: 今はルートが1つでも、将来の分岐用にRouterを挟んでおくことで、後から既存の処理を壊さずに機能追加が可能です。
2. **データ構造を最初に決める**: 後から項目を追加しやすいように、データストアやシートの列定義は少し余裕を持って設計します。
3. **モジュールを分離する**: 巨大な一つのシナリオにするのではなく、機能ごとにモジュール化・連携させます。

待たされるフルコースより、すぐに出てくるサラダ

シェフA（完璧主義）：「完璧なフルコースができるまで出しません」
→ ユーザー（顧客）は去ってしまう。

シェフB（MVP思考）：
「まずサラダを出します。スープはその後です」
→ ユーザーは満足し、メインを待てる。

システム開発も同じ。全機能が完成する半年後まで待たせるより、コア機能を1ヶ月で提供し、使いながら改善していく方が、満足度は高くなります。

完璧主義者への処方箋：マインドとスコープ

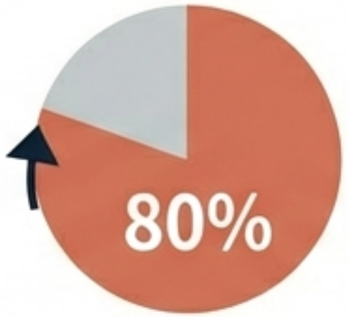
処方箋1：「まず動かす、後で改善」を体験する

理屈ではなく体験が重要です。小さなプロジェクトで「1日で作って公開」し、「不完全でも役に立った」という安堵感を脳に刷り込みましょう。

処方箋2：「やらないこと」を決める

プロジェクト開始時にスコープを文書化する際、「To-Doリスト」と同じくらい重要なのが「Not-To-Doリスト（今回はやらないこと）」です。要望が出ても「それは次フェーズで検討します」と言える準備をします。

完璧主義者への処方箋：自分を縛るルール



処方箋3：80%ルールを設定する

「業務の80%をカバーできたらリリースする」と事前に決めます。残り20%は、当面手動で対応するか、頻度が低いなら対応しないと割り切ります。

処方箋4：タイムボックス（時間制約）を設ける

「この機能は2週間で完了させる」と期限を区切ります。時間が限られていれば、人間は自然と「本当に必要な機能」だけに集中し、「あったら良いな」を削ぎ落とすことができます。

「不完全でも大丈夫」と言える心理的安全性

🗣️ **コンテキスト:** 完璧主義を脱却するには、失敗が許容される環境が必要です。

🤝 **リーダーの役割:** 「80点でOK」と明言し、不完全なリリースを批判せず、そのスピードを評価すること。

💬 **自己対話:** 「これは恥ずかしい仕事だ」という自己批判を、「これは改善のためのスタート地点だ」という前向きな言葉に置き換えましょう。

まとめ：新しいスタンダードへの転換

Old Mindset	New Mindset
One Big Perfect Release	Continuous Small Improvements
Waiting for 100% specs	"We can fix it later" (Agility)
Fear of imperfection	80% Release -> Iterate

- 完璧主義は敵ではなく、コントロールするもの。
- MVP思考で「これ以上削れない」最小単位で価値を届ける。
- ノーコード時代の強みである「後で直せる・足せる」を活かす。
- 「完璧なものを作る」から「価値を届けて改善し続ける」へ。

さあ、80点の勇気を持って、最初の一歩を

完璧なタイミングや、完璧な仕様書を待つのはもうやめましょう。今抱えているプロジェクトのスコープを半分に削り、今週中にリリースできる形にしてみてください。

世界はあなたの「完璧な計画」ではなく、「現在の実行」を待っています。

Done is better than perfect.
（完了は完璧に勝る）