

生成AIを利用したソフトウェア開発における 生産性を向上するプロセスの提案

～品質を維持しつつコードレビューボトルネックを解消～

三菱電機株式会社

三木 将司

開発における問題点

生成AIによって実装は高速化した一方、人手のコードレビューが次のボトルネックになっている。レビュー量が増えると確認がおざなりになって不具合が後工程へ流出し、スキルを持つ人へ負荷が集中する。人のレビュー速度には上限があり、200行/hを超えると欠陥検出率は45～51%まで低下する。

手法・ツールの適用による解決

振舞い駆動開発（BDD）の考え方を採用し、IF設計と同時に「ユーザーの振舞い」からテストを設計し、実装前に契約として固定する開発プロセスを提案。テストが通れば機能正確性は担保されるとみなし、**実装コードに対する機能要件観点のレビューを省略**する。検証の結果、品質を維持したままコードレビュー時間を**39%削減**できた。

提案：テストを実装前に「契約」として固定し、実装コードのレビューを省く

現状の開発プロセス（実業務）



提案の開発プロセス



- ① IF設計と同時にテスト設計
- ② テストを満足するよう実装
- ③ テストOKならレビューを省略

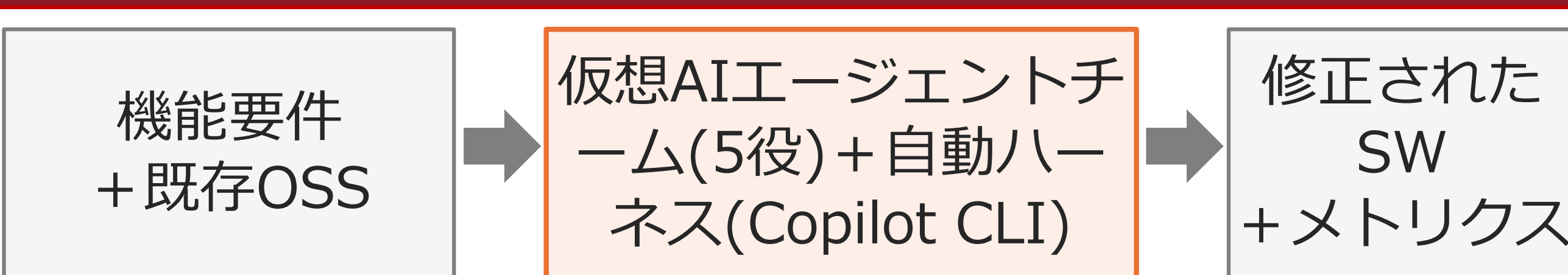
※ 現状プロセスとの差分は振舞い駆動の工程のみ（他工程・プロンプトは同一）

【仮説】

機能要件が明確であれば、機能要件に対するテストを事前に「契約」として実装することで、実装コードに対する機能要件観点のレビューを省略できる

※「テストが通れば正しさが担保される」という判断は、既に開発現場の経験則として観測されている（Dhanorkar et al. 2026）

検証方法

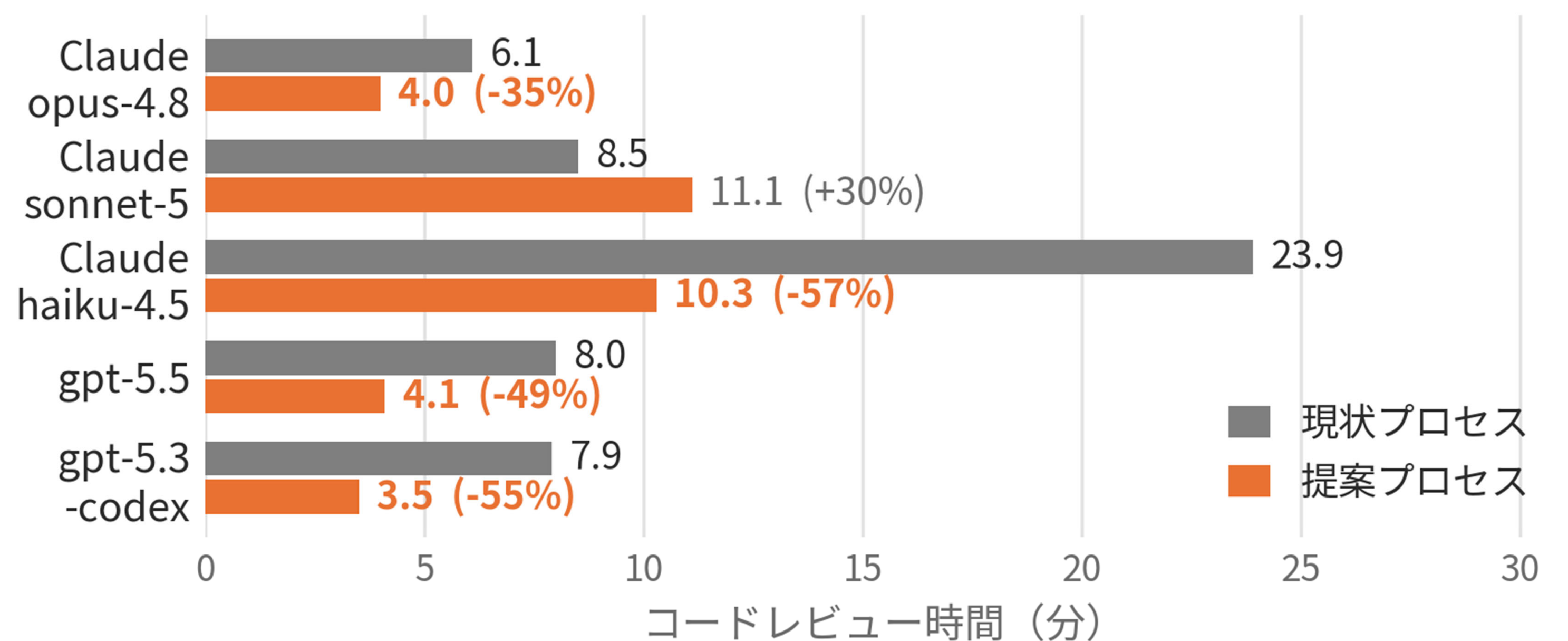


※ 両プロセスで同一の機能要件・同一のエージェントを使用

- ・**評価対象**：Web解析OSS「Umami」に「各サイトにメモを残せる」機能を追加(受け入れ条件7項目)
- ・**サンプル**：5モデル（opus-4.8 / sonnet-5 / haiku-4.5 / gpt-5.5 / gpt-5.3-codex）×2プロセス＝10サンプル
- ・**測定**：①人手レビュー時間（開始からPR承認まで自動計測）②受け入れ条件充足率 ③利用コスト
- ・**ばらつき対策**：飛ばさず全文を読む／事前に試しレビュー／割込み時はやり直し

評価結果

モデル別のコードレビュー時間（現状 vs 提案）



コードレビュー時間 合計 54.4分 → 33.0分 (-39.4%)

全工程のレビュー時間も 44.3分 → 30.8分 (-30.6%)

- ・受け入れ条件充足率は **97.1% → 100%**（品質は維持）
- ・最も速く安いモデルは **gpt-5.3-codex**（23.8分／660円）

まとめ・今後の展望

品質（受け入れ条件充足率 **97.1%→100%**）を維持したまま、コードレビュー時間を**39.4%削減**できた。実業務ではまず振舞い駆動によるテスト設計・自動化から適用し、ハーネス整備とメトリクス測定でプロセス改善とモデル選定を継続的に進める
※ 各条件1サンプルのみ、AIが作成したテスト自体の妥当性は未検証、レビューは著者1名という制約が残る。

スマートエスイー：スマートシステム＆サービスおよびDX推進を担う人材の産学連携育成