投稿日 2025/02/26

GitHub Copilotを活用した大規模開発 ~オフショア開発での実践と知見~

TISでは生成AIを活用した開発を推進しており、その一環としてGitHub Copilotの利用を進めています。
本記事は、大規模開発でのGitHub Copilotの利用に焦点をあてた取り組みをご紹介します。

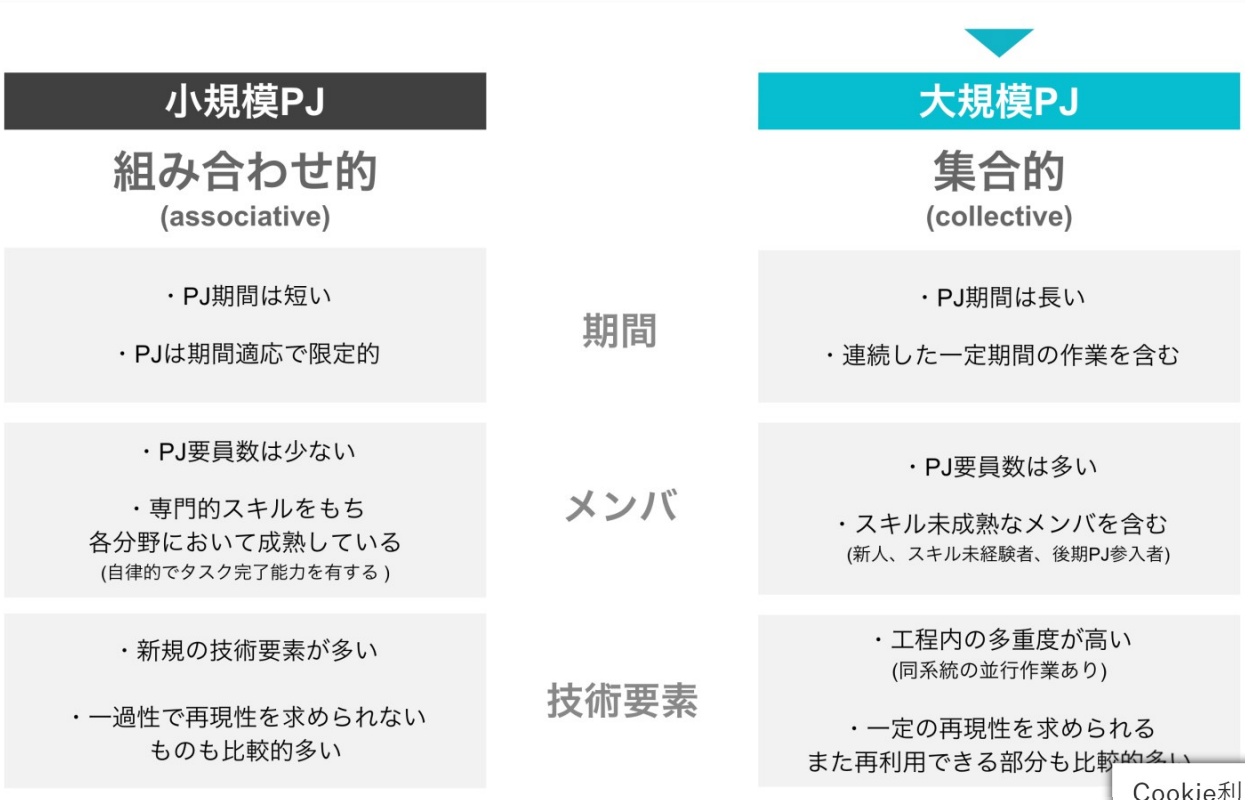
はじめに

TIS 産業モダナイゼーションビジネス部では2024年上期に生成AIを活用したグローバル開発センター(AI-GDCC)のコンセプト検証を行い、GitHub Copilotを活用したオフショア開発の実現性と効果を確認しました。その後、2024年下期よりプロジェクトへの展開を実施しています。

GitHub Copilotの導入にあたって

TISの大規模開発においては、多いケースでは1プロジェクトで100名近いエンジニアが多拠点で並行開発を行います。多拠点開発プロジェクトを安定進行・効率化するために、大規模開発ならではの前提を考慮した導入を検討する必要があります。

開発者間のスキルレベル差が大きく、コードの品質にムラが生じやすい **多人数での「コード品質の維持」をどのように実現するかは、**最上位の留意点になります。



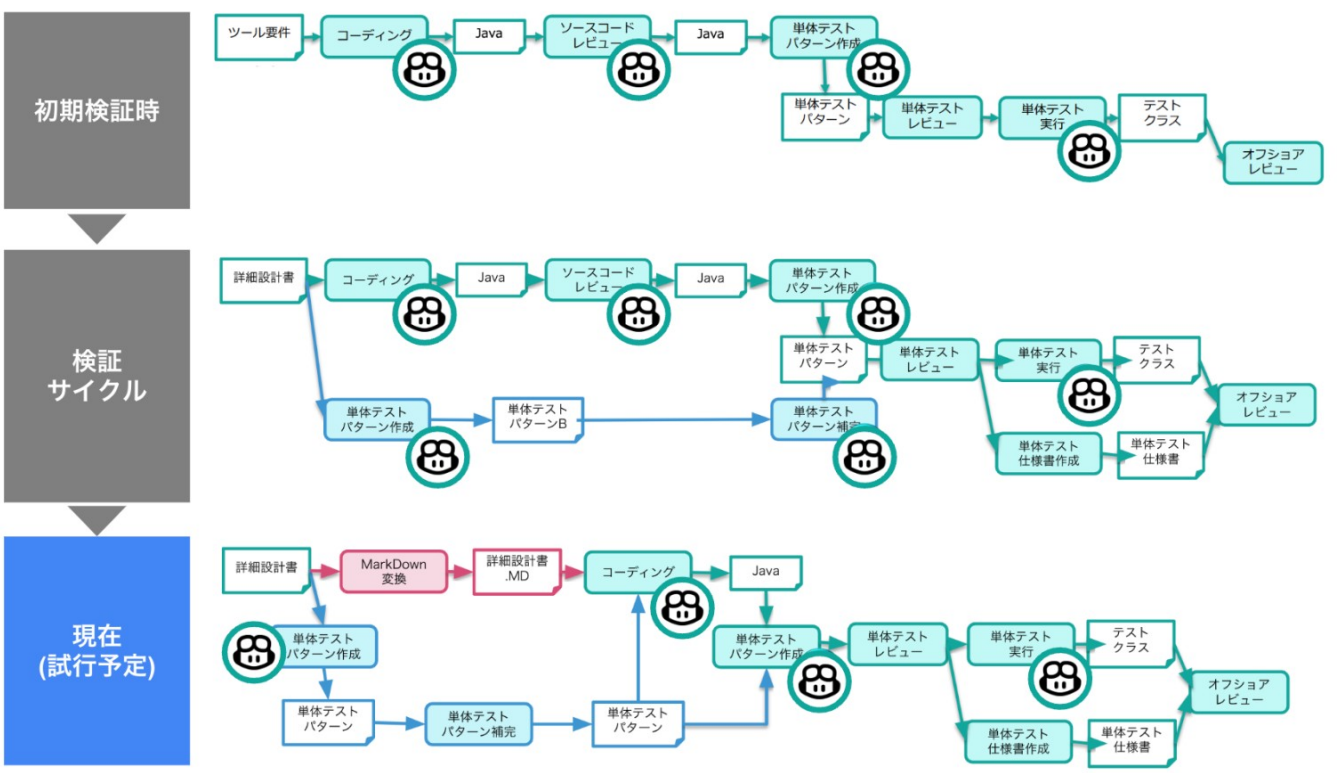
導入推進のポイント

前述の留意点にもとづき、導入計画を進めていきました。

1.GitHub Copilotを用いた開発フローの整備

複数のエンジニアが同一のAIツールを使用するだけではコード品質のばらつきを抑えることができません。同じツールを使用しても、要員ごとにツールの使い方にばらつきが生じてしまうためです。

対策として「**どのアクティビティでGitHub Copilotを使うか/使わないか**」「**開発のアクティビティの順序**」をフローに起こし、プロジェクトの認識をそろえました。これらのワークフローは、利用者側のツール理解とともに段階的に改善していきました。



開発フローのうち、GitHub Copilotを使用するアクティビティを示している

2.GitHub Copilot Chatで使用するプロンプトの標準化

検証期間中にプロンプトエンジニアリングが生産性と品質に大きく影響することが分かりました。以降は使用するプロンプトを予め作成し、メンバーに配布する方式としました。「**どのアクティビティでどのプロンプトを使うか**」までを開発ガイド内で定めたということになります。

3.GitHub Copilot Chatで使用するプロンプトの最適化

①Markdown形式への変換

エクセルやワード形式の資料は、AIにそのまま読み込ませても良い精度が望めません。
従前のコーディング規約のうち、エクセルやワード形式で提供しているものはMarkdown形式へと変換しました。

②ファイルの分割

AIの応答精度を望むには、適切なサイズに分割した依頼が必要になります。
コードレビューに関してはレビュー観点別 （「可読性」「パフォーマンス」等）に、プロンプトを整理しました。

また、コードレビュー用のプロンプトは結果を視認しやすいように表形式で出力されるように統一しました。

```
SQLコードを評価し改善案をmarkdown表形式で示して。表以外は表示しないこと。

## 表
観点、結果(○,×,△,-)、改善点

## rule
* JOINの条件が明確に記述されているか
* NULLの比較に IS NULL や IS NOT NULL を使用しているか
* WHERE句の条件が論理的にグループ化されているか
* 集計関数を使用する際、NULLの扱いが明示的か
* 定数やマジックナンバーを避け、パラメータ化されているか
* サブクエリに適切な名前が付けられているか
```

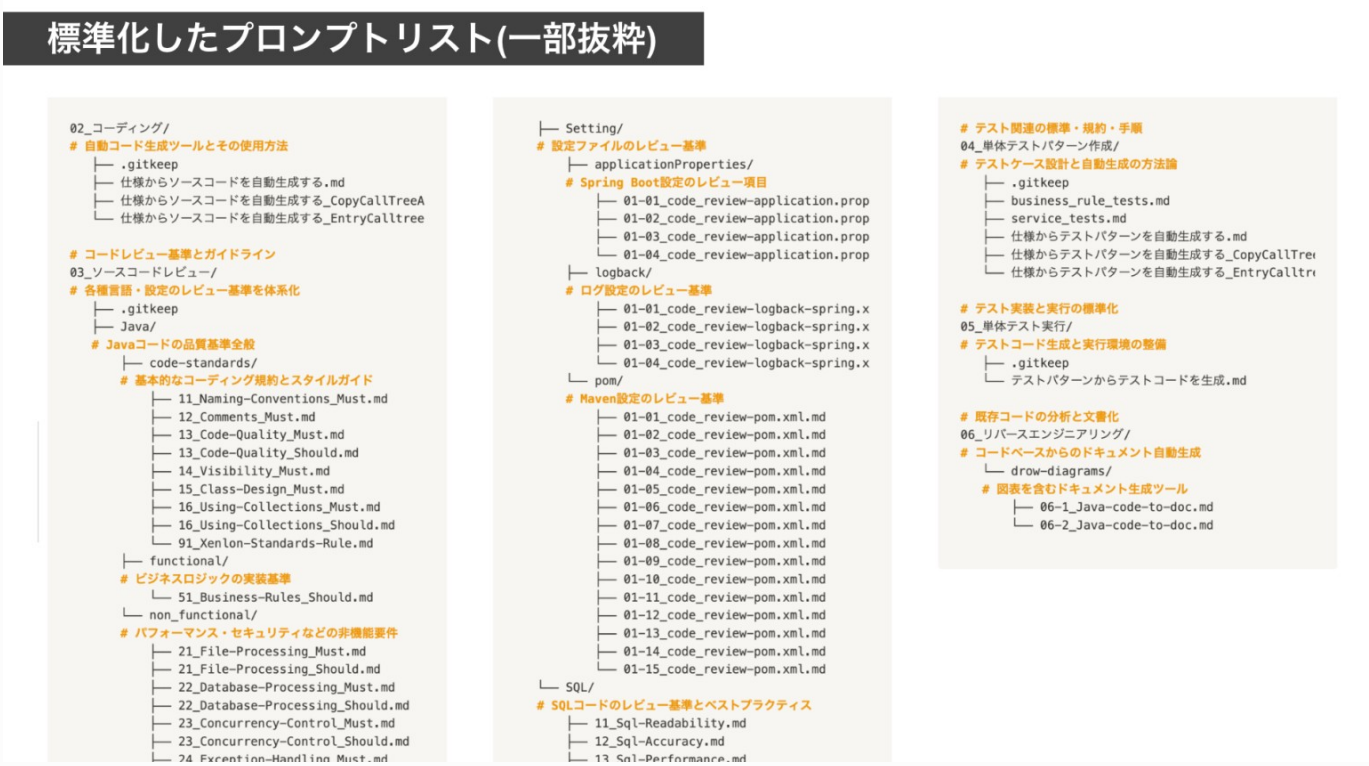
```
SQLコードを評価し改善案をmarkdown表形式で示して。表以外は表示しないこと。

## 表
観点、結果(○,×,△,-)、改善点

## rule
* ORDER BYで数値による列指定ではなく、列名を使用しているか
* インデントが適切に使用されているか
* エイリアスの使用が適切で分かりやすいか
* キーワードは大文字で統一されているか（SELECT, FROM, WHERE等）
* コメントが適切に使用され、複雑なロジックが説明されているか
* テーブル名とカラム名の命名規則が一貫しているか（スネークケース推奨）
* 長いクエリが適切に改行されているか
```

```
SQLコードを評価し改善案をmarkdown表形式で示して。表以外は表示しないこと。

## 表
観点、結果(○,×,△,-)、改善点

## rule
* トランザクション管理のステートメントが適切に使用されているか
* DDL文でデータ型と制約が明示的に指定されているか
* スキーマ名が明示的に指定されているか（適切な場合）
* 一時テーブルやCTEの使用が適切で、名前が説明的か
* 重複したコードや条件を避け、再利用可能な形になっているか
```

SQLコードのレビュー観点別プロンプト（一部抜粋）

③プロンプトのリポジトリ管理

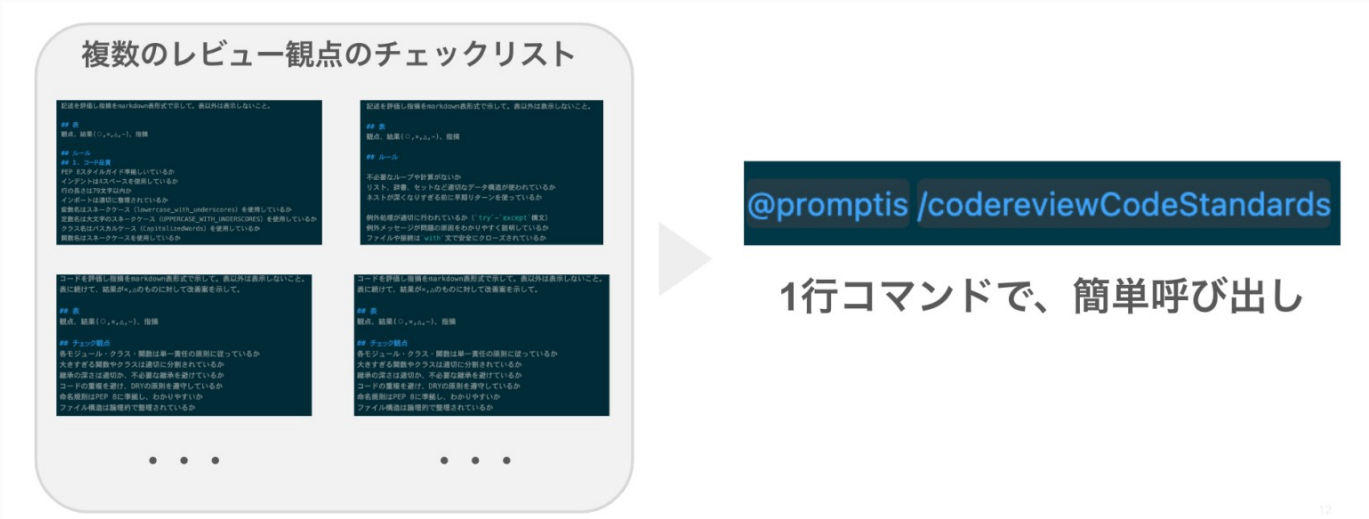
各自が好きにGitHub Copilotを使用する状況は避けるために、プロンプトの構成管理を徹底し、統一的な使用方法の確立を計画しました。レビュー観点の分類を階層型に整理し、体系化を行いました。



標準化したプロンプトの一覧（一部抜粋）

4.プロンプト呼び出しの簡単化

細分化されたプロンプトを一つずつ呼び出すのは手間ですので、コマンドでまとめて呼び出しできるように、Visual Studio Codeエクステンション「Promptis」 を作成し、開発者に利用してもらう方針としました。



「Promptis」はチャットログを指定ディレクトリにバックアップ出力する仕様としており、開発担当

がレビュー結果を共有しやすい工夫もなされています。

GitHub Copilotの導入効果

生産性・品質の効果の総合評価

アクティビティごとの評価は以下の通りです。特にソースコードレビューの効率化により顕著な成果を得ました。（検証では定量評価を行っていますが、本記事では定量値は非公開といたします）

	コーディング	単体テスト	受入テスト
生産性の影響	△ コード補完による作業時間短縮は限定的	◎大幅な作成時間の短縮	◎ オフショアでの品質チェックの実現効率化 ・受け入れ担当の負荷軽減
品質の影響	◎ コードレビューによる非機能面の品質向上	◎網羅的なテスト実現	◯ 製造不具合検出率の向上

受け入れ品質改善による効果

オフショア開発では、TIS側でコードと単体テスト結果の確認を含めた品質チェック（=受け入れテスト）を実施します。

通常の静的解析では検出しづらい非機能面のチェックなどマージ前の品質チェックや単体テストを含めた包括的なレビューをオフショア側で実施可能となったことにより、受け入れ担当の負荷が改善されました。

導入前	導入後
以下の課題があった ・品質の低いコードが頻出すると受け入れ担当の負荷が高まる ・受け入れ担当でコード修正する必要があり、技術者の確保が必要。オフショア人員を増やすと受け入れ体制も増強する必要があった	以下の改善が見られた ・オフショア側で品質チェックの質が改善され、受け入れレビューでの対応コストが削減 ・受け入れ担当の負荷が最適化され、受け入れ体制をそのままにオフショア人員を増やすことが可能になった

まとめ

TISにおける大規模開発でのGitHub Copilotの利用についてご紹介しました。

AIコードアシスタントのメリットを最大限享受するには、**注意点を認識し、AIツール利用を含む開発プロセスそのものを適切にコントロールする**ことが必要です。

大規模開発では適切な手順整備が不可欠です。ツール導入だけでは課題解決に至らず、プロンプト整備プロセスを確立と継続的な改善が必要です。

今後の展開：プロンプト拡充を全社共同のプロセスへ

TISでは今後もGitHub Copilot利用を進めていきます。プログラミング言語・アプリケーションフレームワーク別、コンポーネント別、レビュー観点別の複数の再利用可能なプロンプトの拡充を全社共同のプロセスとして実施していきます。

/* Recommend */

「Generative AI（生成AI）」 のおすすめ記事はこちら

この記事に関連する記事もお読みください。

Generative AI（生成AI） NEW

OctoNihon Forumイベント発表資料「GitHub Copilotを活用した大規模開発の 今 と 未来」公開

2025/06/06 4

Generative AI（生成AI）

インフラコードに対するAIコードレビュー：GitHub CopilotとPromptisの活用

2025/03/26 9

Generative AI（生成AI）

GitHub Copilotを用いたAIによるコードレビューの活用

2025/01/28 26

最近投稿された記事も用意しました。

Generative AI（生成AI） NEW

「金融業界特化型AIエージェントワークショップ」開催レポート

2025/06/13 8

Generative AI（生成AI） NEW

OctoNihon Forumイベント発表資料「GitHub Copilotを活用した大規模開発の 今 と 未来」公開

2025/06/06 4

ITアーキテクチャ NEW

生成AIの新潮流：AIエージェント勉強会を開催しました

2025/05/02 12

「Generative AI（生成AI）」 で最も読まれている記事を以下にまとめています。

NEW

Generative AI（生成AI）

GitHub Copilotを用いたAIによるコードレビューの活用

2025/01/28 26

Generative AI（生成AI）

OctoNihon Forumイベント発表資料「GitHub Copilotを活用した大規模開発の 今 と 未来」公開

2025/06/06 4

Generative AI（生成AI）

GitHub Copilotを活用した大規模開発 ~オフショア開発での実践と知見~

2025/02/26 16