# AI ナビゲーション型フリースタイルテストの提案

## - テスト技術不足を補う，仕様の詳細化不十分に起因するバグの検出手法 -

# Proposal for an AI-Navigated Freestyle Testing Method

## - A Method for Detecting Specification-Related Bugs to Compensate for Insufficient Testing Skills -

リーダー：有森　勇一（株式会社両備システムズ）
研 究 員：津梅　眞嗣（株式会社ゼネラル）
　　早坂　淳　（株式会社日立ソリューションズ・クリエイト）
　　丹羽　優斗（アイホン株式会社）
主　査　：喜多　義弘（長崎県立大学）
副 主 査：秋山　浩一（株式会社日本ウィルテックソリューション）
アドバイザー：西田　尚弘（株式会社日新システムズ）

**研究概要**

本研究は，仕様漏れや曖昧さに起因するバグが，対象ドメインに対する知識不足やテスト技術不足によりテストで十分に検出できないという課題を解決するため，生成 AI を活用した新しいテスト手法を提案する．本手法は，テスト実行中に生成 AI がテスト対象の仕様に応じて，確認すべき観点や操作例を提示する．本手法を適用することで，テスターの経験に依存せず仕様漏れや曖昧さを検出可能であることを実験で確認した．

**Abstract**　This paper proposes a novel testing approach that leverages generative AI to address the challenge that bugs caused by missing or ambiguous specifications are not sufficiently detected during testing due to testers' lack of domain knowledge and testing skills. In the proposed approach, generative AI supports test execution by dynamically presenting viewpoints to be checked, and example operations based on the specifications of the test target. Experimental results confirm that this approach enables the detection of specification omissions and ambiguities, regardless of the tester's experience level.

## 1.　はじめに

### 1.1　研究背景と課題

研究員の職場は 10 人以下の小規模体制のため，開発者が設計や実装に加えてテストも兼任している．運用中のトラブルを分析したところ，表 1 に示すバグ原因の分類定義に基づき整理した結果，図 1 に示すように，全体の 76%が仕様漏れ・曖昧（本稿では，機能そのものの欠落を指すのではなく，入力条件・例外条件・前提状態に対する振る舞いが仕様として十分に定義されていない状態を指す）に起因するバグであった．

バグの見逃し原因を分析したところ，独立した視点での観点抽出が不足していたことが明らかになった．これは，開発者本人がテストを行ったことが原因と推測される．また，過去のバグや業務知識が組織として蓄積できておらず，特定の担当者の経験に依存している実態も確認された．

仕様漏れや曖昧さに起因するバグに対しては，上流工程に遡って仕様定義を改善することが本質的な対策である．しかし，研究員の職場環境では，リソースやコストの制約から，上流工程に対して現状以上の工数や体制を投入することは困難であった．そのため，下流工程の終盤において，有識者による非形式的なテストを追加実施することで対策してきたが，有識者の工数確保には限界があった．

このような環境ではテストが経験に依存し，有識者以外のテスターが仕様漏れや曖昧を十分に発見できない課題がある．開発者がテストを兼任する体制であるため，テストの専門的なスキルが必ずしも備わっているとは限らず，事前にテスト観点を提示していても，異常系の観点や入力値のバリエーションが限定的となりやすい．その結果，仕様に明示されていない条件や想定外の操作を対象としたテストが不十分となる傾向にある．

このような背景から，テスターの経験に依存せず，テスト技術不足を補完しながら，仕様漏れや曖昧を効率的に検出できるテスト手法が求められている．

表 1　バグ原因の分類定義

| 分類 | 定義 |
|---|---|
| 仕様漏れ・曖昧 | 要件は明確であるが、仕様の詳細化が不十分・曖昧 |
| 設計不備 | 仕様は明確であるが、設計上の考慮不足 |
| 実装ミス | 仕様・設計は正しいが、実装が誤っている |
| 環境設定ミス | 導入時のミス（プログラムに問題はなし） |

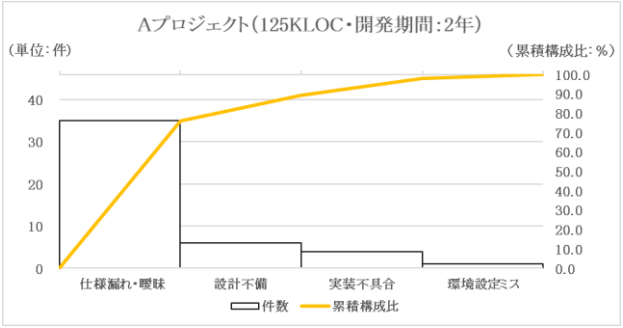


図 1　運用トラブルのバグ原因

### 1.2　提案概要

そこで本研究では，これらの課題を解決する新しいテスト手法として，「AI ナビゲーション型フリースタイルテスト」を提案する．本手法は，生成 AI がテスト実行中に，テスト対象の仕様情報に応じて確認すべき観点や，それを具体化した操作例や入力例を提示し，テスターの行動をナビゲーションする点に特徴がある．これにより，有識者でなくても，仕様漏れや曖昧さに起因したバグを発見できるようになることを目指す．

本論文では，2 章で先行研究の調査結果と解決すべき課題を整理し，3 章で提案手法の具体的内容を述べる．4 章では有効性検証のための実験および結果を示し，5 章で考察を行う．6 章では残課題と今後の展望について述べる．

## 2.　関連研究

### 2.1　先行研究

仕様漏れや曖昧さに起因するバグをテストで検出するための手法について，先行研究を調査した．先行研究は，大きく二つの方向性に分類できる．

一つは，テスト実行前の分析や準備作業を支援するアプローチである．小楠ら[1]は，LLM に仕様書などのテストベースを入力することで，仕様が不十分な個所やバグが潜在しやすい箇所を抽出し，探索的テストにおける着目観点を事前に提示する手法を提案している．この手法は，人手による仕様読解や分析作業の負荷を軽減し，テスト実行前の準備を効率化する点で有効である．

もう一つは，ベテランが有する暗黙知を形式知化し，テスト設計や学習を支援するアプローチである．飯沼ら[2,3]による探索的テストの留意点パターン化やゴモラメソッド，田口ら[4]によるバルタンメソッドでは，ベテランの経験やバグ発見パターンを構造化し，テストパターンマトリクスやプロセスとして定義している．特に，ゴモラメソッドやバルタンメソッドでは生成 AI を活用することで，テスト準備や情報整理が効率化されている．これらの研究は，ベテランの知見を活用することで，観点漏れの防止やテスト品質の向上を目指している点に特徴がある．

### 2.2　解決すべき課題

先行研究は，仕様漏れや曖昧さに起因するバグの検出に対し，テスト実行前の段階で，

着目すべき観点を提示する点で有効な支援を行っている．これにより，経験の浅いテスターでも一定の成果を得られることが報告されている．しかし，これらの手法はいずれも，提示された観点を具体的な操作や入力へ落とし込み，実行できることを前提としている．

一方で，本研究の対象とする職場環境では，テスターが十分なテスト知識を有しているとは限らず，観点が提示されていても，それを具体的な操作へ結びつけることが難しい．そのため,観点提示のみではテスト実行中の行動を十分に支援できないという課題が残る．

先行研究における LLM を活用した探索的テスト効率化や，探索的テストの留意点パターン化，ゴモラメソッド，バルタンメソッドはいずれも，テストの方向性や観点を事前に提示することに主眼が置かれており，テスト実行中に操作を具体的に補助する仕組みは存在しない．そのため，本研究の対象とする職場環境には適さないと判断した．

以上を踏まえ，本研究では，テスト実行段階における仕様漏れや曖昧さに起因するバグの検出を支援するため，観点を具体的な操作例や入力例として示す手法を検討する．その有効性を検証するため，以下の研究課題を設定する．

RQ1：本手法は，経験に依存せず，仕様漏れや曖昧さに起因したバグを検出できるか？

RQ2：本手法にて示す操作例や入力例は，テスト実行において実用的か？

## 3.　提案

### 3.1　提案手法の概要

#### 3.1.1　提案手法の基本概念

本研究では，テスト実行段階において，生成 AI がテスターに対し，確認すべき観点や操作例，入力例を提示し，テスト実行を支援する手法を提案する．

従来，非形式的なテストでは,「次に何を試すか」「どこに着目するか」といった判断は，テスター自身の経験や知識に依存してきた．そのため，経験の浅いテスターでは，観点抽出が不十分になり，期待する効果が得られない場合があると指摘されている[5]．

本手法では，テスト実行中に生成 AI が，本研究で開発した AI プロンプト（以下，テストナビプロンプトという）をもとに，着目すべき観点や試行価値のある操作例，入力例を抽出し提示する．ここで提示する内容は，テストケースや手順として実行を強制するものではなく，テスターが次の試行を考えるための選択肢として利用される．

本手法の目的は，テスト実行の進行や判断を生成 AI に委ねるのではなく，従来は個人の経験に依存していた着眼点や発想の切り口をテストナビプロンプトによって明示化し，テスト実行中の試行を支援するものである．これにより，経験の浅いテスターであっても，試行価値の高い操作に気づきながらテストを進めることが可能となる．

なお，本手法は，ウォーターフォール開発における単体テストからシステムテストまでの工程を一通り実施した後に，非形式的なテストとして実施されていた確認作業を代替する形で，追加テストとして適用することを想定している．そのため，AI が提示する観点や操作例の中には，既存テストで実施済みのものが含まれる可能性がある．この場合，テスターは，既存テストの実施状況と照らし合わせることで，テストの要否を判断する．

#### 3.1.2　先行研究との違いと新規性

先行研究では，テスト観点に関する助言の提示が中心であった．一方，本提案は，テスト実行段階そのものに生成 AI を介在させ，テスト観点に加え，具体的な操作例や入力例を提示する点に新規性がある．

#### 3.1.3　提案手法の適用条件

本研究では，生成 AI として，Microsoft Copilot（無料版，Think Deeper モード）を使用した．本手法は，テキスト入力を受け取り，文脈に基づいた応答生成が可能な LLM であれば適用可能であり，特定のテストツールや高度な事前設定を必要としない．

## 3.2　提案手法の詳細

### 3.2.1　テストナビプロンプトの構造と生成方法

本研究で開発したテストナビプロンプトの構造と生成方法について説明する.

本手法におけるテストナビプロンプトは，テスト実行中に生成 AI が適切なテスト観点や操作例,入力例を提示できるように設計されており,以下の三つの階層から構成される.

(1)制御ルール定義

生成 AI の役割や振る舞い，対話の進行方法を定義する階層であり，本手法の基本的な枠組みとして原則編集を必要としない.

(2)仕様情報

テスト対象システムに関する仕様情報を定義する階層である．画面や機能の概要，前提条件，制約事項など，生成 AI がテスト対象を理解するために必要な情報を本階層にテキストとして記載する．抽象的なシステム概要（例：一般的な販売管理システム）でも入力として扱うことが可能である．ただし，仕様を具体的に与えることで，生成 AI が提示する内容の精度をより高めることができる.

(3)観点・ナレッジ情報

テストにおいて有効とされる観点を定義する階層である．入力データパターンや異常系など，一般的なテスト観点をあらかじめ保持しており，テストナビプロンプトを修正しなくてもテスト実行支援が可能である．さらに，対象システム固有の条件や運用上の注意点，過去のバグ情報などを，本階層の最下行にテキストとして追記することで，生成 AI による観点提示の精度を向上させることができる.

本手法を第三者が適用する際には，これら三つの階層のうち，主に編集が必要となる箇所が異なる．表 2 に各プロンプト階層の編集要否と編集方針を示す.

**表 2　第三者が本手法を適用する際のテストナビプロンプト編集指針**

| プロンプト階層 | 主な役割 | 編集要否 | 編集方針 |
|---|---|---|---|
| 制御ルール定義 | ロール定義・進行制御 | 不要 | 本手法共通の枠組みとして固定 |
| 仕様情報 | テスト対象の理解 | 必要 | 対象システムに応じて記載内容を置き換え |
| 観点・ナレッジ情報 | 観点展開の補助 | 任意 | 業務特性や過去事例に応じて追記・調整 |

### 3.2.2　テストナビプロンプトを用いたテスト実行プロセス

本手法の実行プロセスについて,準備段階とテスト実行段階の 2 段階に分けて説明する.

(1) 準備：テストナビプロンプトの作成

本手法は，テストリーダーが事前に作成したテストナビプロンプトを起点として実行される．テストナビプロンプトは，本研究で作成した汎用的なプロンプト（付録 1 に記載）にテスト対象の仕様情報をテストリーダーが付加して作成する．現状では，プロンプトにはテキスト情報のみを用いている.

(2) テスト実行段階：AI がテスト実行者に対してナビゲーション支援

テスト実行時には,テストナビプロンプトをテスターが生成 AI に与えて起動し,テストナビゲーションを開始する．テスターはテストナビプロンプトを修正したり，追加の情報を入力したりする必要はない.

生成 AI は，起動後，テストナビプロンプト内の情報を解析し，テスト対象とすべき機能群を抽出して提示する．テスターは提示された候補からテスト対象機能を選択する．機能が確定すると，生成 AI は，当該機能に対し，仕様漏れや曖昧さが生じやすいと考えられるテスト観点や操作例を一覧として提示する．これらの観点には，正常系，異常系，

境界条件に加え，過去のバグ傾向や運用上の注意事項と関連の深い振る舞いが含まれる．

テスターは，提示された観点や操作例を参考に，既存のテストで未確認の箇所を中心にテストを行う．また，必要に応じて，生成 AI と対話し，観点の追加提示やテスト対象機能の切り替えを指示してテストを継続する．

### 3.2.3　テストナビプロンプトの特徴

本研究で開発したテストナビプロンプトは，テスト実行を支援するために設計されたものであり，出力形式や情報構成に特徴を持つ．

第一に，生成 AI との対話を番号入力による指示方式に限定し，出力は見出しや表を用いた Markdown 形式に固定している．これにより，生成 AI の応答形式のばらつきを抑え，テスターが対話方法に迷うことなくテストに集中できる．

第二の特徴は，仕様情報と観点・ナレッジ情報を分離して与える構成とし，生成 AI が仕様理解と観点参照を両立できるようにしている．仕様情報には，テスト対象システムの機能概要や前提条件を記述し，観点・ナレッジ情報には，一般的なテスト観点や，対象システム固有の条件や運用上の注意点，過去のバグ傾向などを記載する．これにより，生成 AI は仕様情報に基づいてテスト対象を理解しつつ，観点・ナレッジ情報を参照することで，試行価値の高いテスト観点や操作例，入力例を提示できる設計としている．

図 2 に，テストナビプロンプトの記述イメージと，生成 AI 実行時の出力イメージを示す．また，付録 2 に，生成 AI 実行時の実際の出力画面のスクリーンショットを示す．

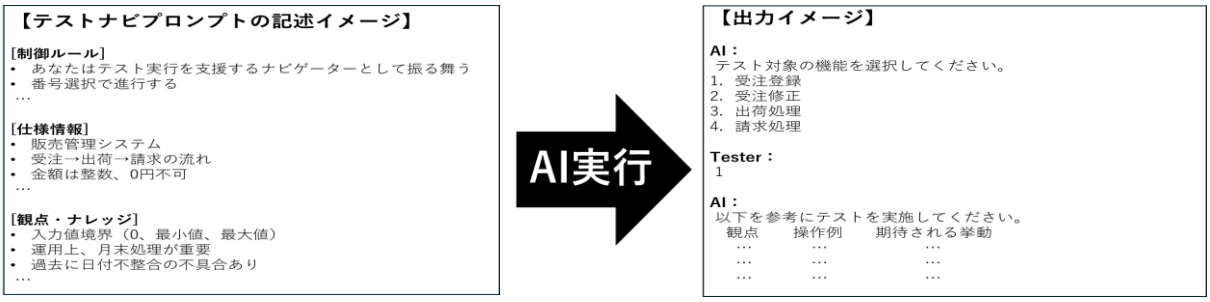


図 2　テストナビプロンプトの記述イメージと出力イメージ

## 4.　評価

### 4.1　実験内容

#### 4.1.1　実験の目的

本実験の目的は，生成 AI がテスト実行時に観点や操作例を提示することで，テスターの経験に依存せず，仕様漏れや曖昧さに起因するバグを検出可能か検証することである．

#### 4.1.2　実験概要

本実験では，同一のテスト対象に対して，アドホックテストと提案手法を順に実施した．その結果，アドホックテストでは検出できなかった仕様漏れや曖昧さに起因するバグを，提案手法によってどの程度検出できたのかを評価した．

#### 4.1.3　実験参加者

本実験には，計 23 名が参加した．参加者はテスト実務経験に基づいてキャリアを分類しており，各キャリアの定義および参加人数を表 3 に示す．

表 3　実験参加者のキャリア定義と参加人数

| キャリア | 定義 | 参加人数 |
|---|---|---|
| 新人 | 実務経験が 0〜2 年のテスター | 9 名 |
| 中堅以上 | 実務経験が 3 年以上のテスター | 14 名 |

### 4.1.4　テスト対象

本実験で用いた環境を以下に示す.

・疑似 SNS 型アプリケーション（テスト対象）
・テスト対象の仕様書
・生成 AI（Microsoft Copilot 無料版 Think Deeper モード）

テスト対象は短文の投稿および閲覧が可能な疑似 SNS 型アプリケーションであり，検証目的のため仕様バグを意図的に埋め込んだ．本実験では，仕様に記載されていない条件や操作を実施した際に，出力や結果の妥当性を判断できない挙動を「仕様バグ」と定義する．また，付録 3 に，埋め込んだ仕様バグの事例を示す.

### 4.1.5　実験方法

本実験では，Web ブラウザから利用可能な Microsoft Copilot を使用したため，特別な環境構築は不要であった．実験開始前に，実験の進め方およびテストナビプロンプトの使用方法を記載した手順書を配布し，参加者はこれを参照して実験を実施した．実験中に質問はなく，滞りなく進行した.

参加者は，同一のテスト対象に対して，以下の 2 種類のテストについて，順序を固定して実施した．各テストの実施時間はそれぞれ 60 分以内とした.

(1) アドホックテスト
テスト設計は行わず，仕様の漏れや曖昧さが生じやすい箇所に着目して，テスターが個人の判断で自由にテストを実施する.

(2) 提案手法（AI ナビゲーション型フリースタイルテスト）
本研究で開発したテストナビプロンプトを用い，生成 AI が提示するテスト観点や操作例を参照しながら，テストを実施する.

## 4.2　実験結果

### 4.2.1　バグ検出件数

検出されたバグの内容を整理するため，「仕様バグ」と「対象外（操作性などの改善要望や仕様の誤認）」の 2 種類に分類した．本研究では，仕様漏れや曖昧さに起因するバグの検出支援効果を評価することを目的として，分析の中心を「仕様バグ」とした.

この分類に基づき，キャリア別の平均バグ検出件数を表 4 に示す.

**表 4　実験結果　キャリア別のバグ平均検出件数**

| キャリア | アドホックテスト | | 提案手法 | | 合計 | |
|---|---|---|---|---|---|---|
| | 仕様バグ | 対象外 | 仕様バグ | 対象外 | 仕様バグ | 対象外 |
| 新人 | 2.7 件 | 3.2 件 | 4.2 件 | 4.6 件 | 6.9 件 | 7.8 件 |
| 中堅以上 | 3.8 件 | 4.4 件 | 2.1 件 | 1.5 件 | 5.9 件 | 5.9 件 |

表 4 より，アドホックテストにおける仕様バグの平均検出件数は，新人が 2.7 件，中堅以上が 3.8 件であり，新人は中堅以上の約 71%であった．一方，アドホックテストと提案手法の検出件数を合計すると，新人は 6.9 件，中堅以上は 5.9 件となり，新人が中堅以上を上回った（約 117%）.

次に，提案手法によって検出された仕様バグの内容を把握するため，ユーザ利用への影響度に基づく分類を行った．支障がないものを「軽微」，支障が生じる可能性があるものを「影響あり」とした．その結果を，図 3 に示す.

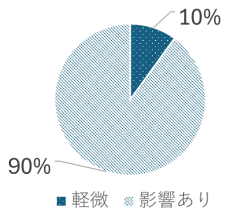


**図 3　実験結果　提案手法によって検出された仕様バグの影響度比率**

図 3 に示すとおり，検出された仕様バグの 90%が「影響あり」に分類された．

また，提案手法によって検出された仕様バグの事例として，その内容，発見の契機となったテスターの操作，および AI 提案に寄与したプロンプト記載を表 5 に示す．表 5 に示した事例以外の仕様バグについては，その一部を付録 4 に示す．

**表 5　提案手法によって検出された仕様バグの事例**

| 仕様バグの内容 | 発見の契機となったテスターの操作 | AI 提案に寄与したプロンプト記載 |
|---|---|---|
| 特定の絵文字が 2 文字分としてカウントされ，文字数オーバーが発生する（サロゲートペア考慮漏れ） | 投稿欄に，AI が提示したコピー&ペースト用の絵文字(連結絵文字)を入力 | 観点・ナレッジ層において，「入力データパターン」として絵文字や結合文字が定義されていたため |

### 4.2.2　アンケートによる提案手法の評価

提案手法の有効性と改善点に関するアンケート回答結果（回答人数：新人 9 名，中堅以上 14 名）を表 6 および表 7 に示す．

**表 6　提案手法の有効性に関する回答結果**

| 意見（自由回答） | 新人 | 中堅以上 | 合計 |
|---|---|---|---|
| 自分では気づかない観点を提示してもらえた | 88% | 86% | 87% |
| テスターの経験依存を低減できると思う | 44% | 43% | 43% |
| テスト効率が向上すると思う | 56% | 29% | 39% |
| テスト中の実施手順の提案が助かった | 56% | 14% | 30% |

**表 7　提案手法の改善点に関する回答結果**

| 意見（自由回答） | 新人 | 中堅以上 | 合計 |
|---|---|---|---|
| 実施できない操作が提案される | 44% | 79% | 65% |
| 生成 AI 提案の妥当性確認が必要 | 56% | 29% | 39% |
| テスト観点に優先順位をつけてほしい | 33% | 43% | 39% |
| テスト観点の不足・過剰がある | 22% | 50% | 39% |

## 5.　考察

### 5.1 提案手法の有効性

本実験では，提案手法がテスターの経験に依存せず仕様バグの検出を支援できることを表 5 のとおり確認した．

アドホックテストでは新人の平均検出件数は中堅以上の 71%であったが，提案手法併用時には 117%となった（表 4）．さらに，対応のある t 検定により，全実験者の平均検出件数は 3.35 件から 6.30 件へ有意に増加しており（$p = 6.8E\text{-}06 < 0.05$），統計的有意性が示された．これにより，新人の検出能力向上と経験差の低減が確認できた．また，提案手法で検出された仕様バグの 90%はユーザ利用に支障が生じる可能性がある「影響あり」に分類された（図 3）．これは，本手法が利用者影響を伴うバグの検出に有効であることを示している．以上の結果は，提案手法が経験に依存せず仕様漏れや曖昧さに起因するバグを検出できるかという RQ1 に対して肯定的な結果を与えるものである．

本手法は経験に依存せず仕様漏れや曖昧さに起因するバグの検出を支援する点で先行研究と整合する．一方で，先行研究がテスト準備段階の支援に主眼を置くのに対し，本手法はテスト実行時に生成 AI が具体的な観点や操作例，入力例を提示する点に特徴がある．アンケートでは，全参加者の 87%が「自分では気づかない観点を提示してもらえた」，新人参加者の 56%が「テスト中の実施手順の提案が助かった」と回答している（表 6）．これらの結果は，RQ2 において操作例や入力例の実用性が一定程度確認されたことを示す．

### 5.2 実験結果に基づく実用上の課題

一方で，アンケート結果（表 7）から，生成 AI の提示内容には改善の余地があることが明らかになった．特に，「実施できない操作が提案される」は，中堅以上の 79%が改善点として指摘した．これは，生成 AI が一般的な観点を優先し，対象仕様の制約を十分に反映できていないことに起因すると考える．実施不可能な操作の提示はテスト思考を停滞させ，生成 AI への信頼低下を招く可能性があるため，操作可能範囲に沿った提案を導くプロンプト設計の改善が必要である．これらは，RQ2 の観点から見た実用上の課題である．

また，本実験では中堅以上の効果が新人より低く出たが，現時点ではその要因を特定できていない．今後，経験別や年齢層別，さらに別のテスト手法も含む追加実験を行い，バグ発見傾向や操作手順の観点で本手法を評価する必要がある．

### 5.3 支援可能範囲に関わる課題

本研究は疑似 SNS 型アプリを対象としており，他ドメインへの適用可能性は今後の検証が必要である．また，本手法では観点に重要度や影響度が付与されないため，実務での優先判断が困難である．そのため，重要度や影響度に基づく優先付けや過去障害事例との関連付けによって，観点の重要性を認識しやすくする改善が必要である．さらに，テスターが開発者の場合，仕様範囲を独自に限定し，本来仕様漏れの事項を対象外と扱う可能性がある．その結果，AI が提示した重要観点が安易に除外される恐れがある．見落としを防ぐ判断指針や対象外と判断した事項を事後に確認できる仕組みの整備が必要である．

加えて，本手法は明示された仕様に基づき観点展開を行うため，要件や機能自体の欠落を導出する仕組みは備えていない．今後は，要件定義書・設計書・テスト仕様書・ソースコード等を横断参照し，文書間の不整合や矛盾を抽出する仕組みの構築が課題である．

さらに，本手法で検出された仕様不備は，上流工程へのフィードバックや観点ナレッジの蓄積に活用可能であり，仕様改善に還元する品質改善サイクルの構築も課題である．

## 6.　おわりに

本研究では，小規模体制下でテストが経験に依存し，仕様漏れや曖昧さに起因するバグを十分に検出できないという課題に対し，「AI ナビゲーション型フリースタイルテスト」を提案した．実験の結果，バグ検出件数は統計的に有意に増加し，新人の検出能力向上と経験差の低減が確認され，RQ1 に対する肯定的な結果を与えた．

また，具体的な操作例や入力例を提示する点は RQ2 で一定程度実用性が確認された．一方で，プロンプト改善による提示内容の精度向上，他ドメインへの適用可能性，優先判断や重要観点の認識を支援する運用面の整備，および得られた知見の上流工程への活用と品質改善サイクル構築など，複数の課題が残る．今後は，これらの課題への対応を通じて手法を洗練することを検討する．

## 参考文献

[1] 小楠聡美，佐々木健人，“LLM を活用した探索的テスト効率化の提案”，ソフトウェア・シンポジウム 2025 in 岡山，2025.
[2] 飯沼真一，“探索的テストを効果的に行うための留意点のパターン化”，ソフトウェア品質シンポジウム 2022，2022.
[3] 飯沼真一，“生成 AI を活用した探索的テストの学習基盤の構築”，ソフトウェア品質シンポジウム 2024，2024.
[4] 田口真義，飯沼真一，“生成 AI を活用したテストパターンマトリックスを用いたテスト観点漏れ低減の提案”，ソフトウェア品質シンポジウム 2024，2024.
[5] 飯泉紀子，鷲崎弘宜，誉田直美[著]，SQuBOK 策定部会[編]，“ソフトウェア品質知識体系ガイド- SQuBOK Guide V3 - 第 3 版”，オーム社，pp.204-205，2020.