生成 AI を活用したテスト設計に関する考察

益子　なるみ
株式会社 JSOL
mashiko.narumi@jsol.co.jp

町田　欣史
株式会社 NTT データグループ
Yoshinobu.Machida@nttdata.com

要旨

ソフトウェア開発における AI 活用の研究や事例が増加し，開発プロセス全体への本格的な活用が期待されている．本稿では，ソフトウェアテストへの生成 AI の活用に焦点を絞り，テストプロセスの中でもテスト設計，すなわち自然言語で記述された要件や仕様からテストケースを作成する作業において，生成 AI サービス ChatGPT（OpenAI 社）を用いることを試みた．ChatGPT に与える指示，すなわちプロンプトによって，特定の観点に対するテストケースが適切に出力されるかを検証し，テスト設計における ChatGPT の有効な利用方法を提案する．

1. はじめに

筆者の所属する組織では大規模なエンタープライズシステムをウォーターフォール開発で開発することが多いが，最近ではアジャイル開発を採用する機会も増えている．アジリティを優先する開発においては，テストを効率的に行うことが必要となる．テストプロセスの中でもテスト実行はテストツールを使って自動化することで効率化を図ることが一般的になっているが，テスト設計，すなわちテストケースを作成する作業については，テストベースを基に人が考えて行っている．そこで，生成 AI の活用に着目し，テスト設計を効率よく行えないかと考えた．

ソフトウェア開発への生成 AI の活用については，多くの研究[1][2]や事例がある．例えば，プログラムコードやコメントからテストコードを生成したり，期待結果とテスト結果の画像を比較したりする作業の自動化に活用されている．しかしながら，生成 AI が出力した結果の品質や精度はまだ低く，単純な計算さえも誤ることがある．また，生成 AI の特徴として，同じ入力に対して異なる結果を返したり，虚偽の結果を返したり（ハルシネーション）することがある．このような問題や特徴を踏まえて，自然言語で記述された要求や仕様からテストケースを生成することの可能性を見極め，有効に活用するための工夫点や注意点を探ることで，現時点で推奨される活用方法を提案する．

2. 現状の課題と解決案

2.1. テスト設計における課題

テスト設計においてテストケースを作成する作業に関しては，テストベースとなる仕様書や設計書を人が確認しながら行うことが一般的であり，まだ自動化が進んでいないため多くの時間を要している．また，人が作業するため，作成したテストケースには抜け漏れが発生することがあり，テストが不足したために重大な問題を引き起こすこともある．

これらの課題を解決するための手段として生成 AI の活用が考えられる．生成 AI を活用することで人の作業が削減されるため，テスト設計の時間短縮と網羅性の高いテスト設計を同時に実現することが期待される．ただし，ソフトウェア開発において生成 AI はまだ導入段階であり，いくつかの課題がある．

2.2. 生成 AI に関する課題

生成 AI の課題としては，まず出力結果の回答精度がある．生成 AI を使って一般的な質問や検索をする場合でも，期待しない結果や明らかに誤った結果が返ってくることがあり，テスト設計に用いた場合でも同様の問題が起こり得る．誤ったテストケースや不十分なテストケースを用いた結果，品質の低いシステムをリリースすることがあってはならない．したがって，その活用には十分な注意を払う必要がある．

その他には，同じ質問をしてもタイミングによって異なる回答をすることから再現性に乏しいため，テスト設計の対象が複数存在する場合に，均質なテスト設計ができない可能性がある．さらには，著作権の侵害や不適切な用語の混入といったコンプライアンスに関する問題もある．

2.3. 解決案

本稿では，生成 AI の回答精度の課題に焦点を当て，生成 AI を用いたテストケース生成における出力の傾向を探り，どのような指示をすればより有効なテストケースが生成できるのかを検証する．先行研究[3]において生成 AI を用いたテスト観点の

導出を提案されているが，これをテストケース生成に活用する．例えば，入力にあるべき例外条件が要件に含まれていなくても，適切なテストケースを引き出すことができれば抜け漏れ防止にも役立つと考えた．

なお，テスト設計におけるもう一つの課題である時間短縮については，作業時間や生産性などの厳密な測定は行わず，精度に関する検証を通して得られる定性的な情報からの分析にとどめる．

3. 検証方法

3.1. 活用方法の提案

基本的にチャット型の生成 AI は対話しながら情報を引き出していくため，テスト設計においても質問を繰り返しながら探索的にテストケースの完成度を高めていくのが理想的と考える．複数のテスト設計担当者により行う場合，質問の方法が担当者のスキルに依存するため，最終的なテストケースの完成度に違いが生じる可能性が高い．そのため，本来は生成 AI との対話プロセスを標準化や定型化することを目指したいが，回答が都度変わる可能性があるため難易度が高いと考えた．そこで本稿では，1回（最初）のプロンプトの指示でできるだけ完成度の高いテストケースを作成することを目指し，最適なテストケースを出力するためのプロンプトの書き方を提案する．

3.2. 検証に使用したサービス

検証に使用した生成 AI は OpenAI 社が提供している ChatGPT-3.5 である．このバージョンは無料で利用できるものであるが，学習データが 2021 年 9 月までと情報が古く，有償版である GPT-4 より LLM の回答精度が劣るといった制限がある（2024 年 3 月 12 日現在）．

なお，本検証では同一条件下で精度を確認するため，再学習不可とした状態で利用した．ChatGPT に与える質問，すなわちプロンプトを入力し，出力結果を得られたらプロンプトをクリアして，新しいチャットの状態としてから次の検証を実行した．ただし，ChatGPT がすべての結果を一度に出力しきれなかった場合のみ，結果の続きを確認するために表示される「Continue generating」を利用して，すべての結果を出力した．また，この条件でテストケース生成を行った場合でも，同じプロンプトに対して異なる回答が出力されることがあるため，1つのプロンプトに対するテストケース生成は複数回（3 回）行い，最もよい結果を採用した．

なお，本検証の「精度」とはテストケースとしての精度という意味であり，書籍の解答例との比較やテスト設計スキルの高い技術者のチェックにより精度を評価する．

3.3. テスト設計の対象

テスト設計の対象は先行研究[3]でも使用されている「マイヤーズの三角形問題」[4]とした．本検証では，アジャイル開発におけるバックログの記法として一般的に用いられるユーザーストーリーの形式で要求仕様を記述し，ChatGPT に入力するプロンプトとした．このユーザーストーリーは「三角形の種類判定システム」を構築するための仕様を ChatGPT で作成したものである．ユーザーストーリーとした理由は「マイヤーズの三角形問題」の機能をもつ1つのアプリケーションを想定することでエンタープライズシステムを開発する筆者の所属組織で実際に活用するイメージに近いものが生成できると考えたからである．

プロンプトの基本的な構成を図 1 に示す．プロンプトは，ユーザーストーリーからテストケースを作成することの指示を記述するプロンプト指示部と，具体的な要求仕様を記述するユーザーストーリー部，出力形式を定義する出力形式部から成る．

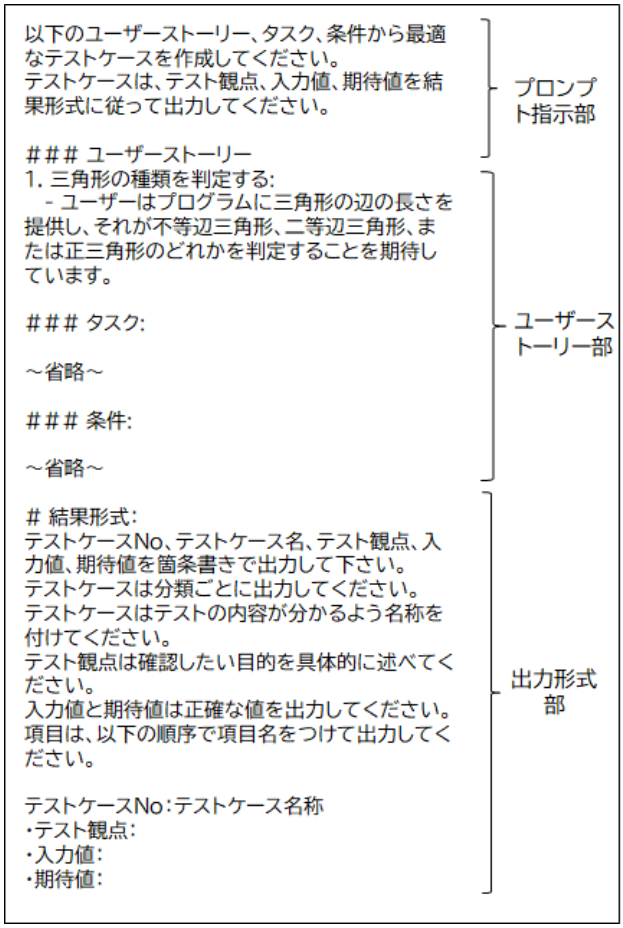

図 1 プロンプトの基本的な構成

プロンプト指示部は 1 文目ではどのようなテストケースを作成

して欲しいかを指示し，2 文目では出力結果に含めて欲しい情報を指示する．

一方，ユーザーストーリー部には，図 1 に示す三角形の種類判定に関する要求に加えて，タスクと条件を記述する箇所がある．タスクには主に機能に関する記述，条件には主に品質に関する記述をする．表 1 と表 2 には，基本となるプロンプト（以降，基本プロンプトと呼ぶ）における 4 つのタスクと 5 つの条件を記載している．いずれも項目に続いてタスクや条件を箇条書きで記す形式となっている（実際の入力イメージは付録 1 を参照）．

表 1 ユーザーストーリー部の記述内容（タスク）

項目／タスク
入力受け付け機能
• プログラムはユーザーから三角形の辺の長さを受け取る入力機能を提供する． • エラーハンドリングを行い，不正な入力があった場合に適切なエラーメッセージを表示する．
三角形の種類判定機能
• プログラムは受け取った辺の長さに基づいて三角形の種類を判定する機能を実装する． • 判定はマイヤーズの三角形問題に準拠する．
結果表示機能
• プログラムは判定結果をユーザーに適切に表示する機能を提供する． • 判定結果はわかりやすく，理解しやすい形式で表示される．
処理時間短縮機能
• プログラムは処理時間を短縮するために効率的なアルゴリズムを実装する．

表 2 ユーザーストーリー部の記述内容（条件）

項目／条件
正確性
• 判定はマイヤーズの三角形問題に基づいて行われる必要がある． • 結果は正確である必要がある．
迅速性
• 判定プロセスは迅速に行われる必要がある． • ユーザーは待ち時間なく結果を受け取ることができる．
使いやすさ
• プログラムは使いやすいインターフェースを提供する必要がある． • 入力や結果の表示が分かりやすくなければならない．
拡張性
• プログラムは将来的な拡張を容易にする設計が必要である． • 新しい機能や特性の追加が容易である．
エラーハンドリング
• 入力が不正な場合，適切なエラーメッセージを表示する必要がある．

3.4. 期待するテストケース

生成 AI を使用した時に出力されることが期待されるテストケースは，マイヤーズが示した解答例[4]を基に，ユーザーストーリー部に記述した表示結果，迅速性，使用性等が追加されたものである．その上で，必ず出力して欲しいものを必須，システムリスクがあまり高くない場面を想定しているために少し冗長や過剰と感じるテストケースを任意として，それぞれ表 3，表 4 に示す．なお，3 辺の順列に関するテストケースや処理に直接影響しない拡張性は期待するテストケースからは割愛した．本テストケースは最終的に品質・テストプロセスの第一人者にレビューによってレビューを行っている．

表 3 生成を期待するテストケース（必須）

No.	テスト観点	入力値	期待結果
入力受け付け機能のテスト			
1	入力値エラー	数字以外	エラーメッセージが表示される．
2	入力値エラー	負の値	エラーメッセージが表示される．
3	入力値エラー	ゼロの値	エラーメッセージが表示される．
三角形の種類判定機能のテスト			
4	正三角形	3 辺の長さが全て等しい	正三角形と判定される．
5	二等辺三角形	2 辺の長さが等しい	二等辺三角形と判定される．
6	不等辺三角形	3 辺の長さが全て異なる	不等辺三角形と判定される．
7	三角形の成立条件エラー	2 辺の和が他の 1 辺よりも小さい．	エラーメッセージが表示される．
8	三角形の成立条件エラー（境界値）	2 辺の和が他の 1 辺と等しい．	エラーメッセージが表示される．
結果表示機能のテスト			
9	正常時表示	三角形判定結果	判定された三角形の種類が表示

No.	テスト観点	入力値	期待結果
			される.
10	エラー表示(エラーハンドリング)	入力や判定，処理エラーの結果	入力や判定エラーメッセージが表示される.
性能要件に関するテストケース			
11	単性能	正常入力値	1処理時間が許容範囲内
使用性に関連するテストケース			
12	ユーザーフレンドリーな表示	—	エラーを含む処理結果が分かりやすいことに言及
13	ユーザーフレンドリーな入力・操作	—	入力操作などの操作性について言及

表 4 生成を期待するテストケース(任意)

No.	テスト観点	入力値	期待結果
入力受け付け機能のテスト			
14	入力値エラー	すべてがゼロの値	エラーメッセージが表示される.
15	入力値エラー	入力する数値の個数が間違っている	エラーメッセージが表示される.
16	正常	数字	入力された辺の長さが正確に受け入れられる.
17	エッジ	小数値や数値型の最大値など	入力された辺の長さが正確に受け入れられる.
18	境界値	境界値を指定	境界内の場合は正常，境界外はエラーメッセージ
性能要件に関するテストケース			
19	大量データ	連続して複数件	正常に三角形を判定して結果が表示される.
20	負荷テスト	複数同時入力	正常に三角形を判定して結果が表示される.
セキュリティに関するテストケース			
21	SQL インジェクション，XSS	SQL やスクリプトタグを含んだ文字	エラー処理，サニタイジングされている.

3.5. 検証の観点

まず，3.3 で示した基本プロンプトを用いた場合，表 3 に示す 13 個の必須テストケースのうち，以下に示す 8 個のテストケースが生成された(実際の出力イメージは付録 2 を参照).

- 入力受け付け機能のテスト … No.2, No.3
- 三角形の種類判定機能のテスト … No.4, No.5, No.6
- 結果表示機能のテスト … No.9, No.10
- 性能要件に関するテストケース … No.11
- 使用性に関するテストケース … なし

生成されなかったテストケースには，以下のものがある.

- 1 つの辺が数値以外である(No.1) .
- 2 つの辺の和が他の 1 つの辺と等しい(No.7).
- 2 つの辺の和が他の 1 つの辺よりも小さい(No.8).
- 使用性に関するテストケース(No.12, No.13).

この結果を基に次に示す 4 つの検証観点を設定した.

(1) テスト技法(エラー推測)

生成されなかったテストケースのうち，初めの 3 つについてはテスト技法の 1 つであるエラー推測を用いて導出される可能性がある. そこで，エラー推測の観点を明示的に指示することにより，期待するテストケースが生成されるか否かを検証した.

(2) テスト技法(境界値分析)

生成されなかったテストケースらのうち，「2 つの辺の和が他の 1 つの辺と等しい」については，テスト技法の 1 つである境界値分析を応用すれば導出が可能であると考える. そこで，境界値分析を用いることを明示的に指示することにより，期待するテストケースが生成されるか否かを検証した.

(3) テストタイプ

テストタイプに関しては，ユーザーストーリーの条件として指定した正確性と迅速性に対応するテストケースは生成されたが，使いやすさ(使用性)については生成されなかった. また，ユーザーストーリーに記述がない品質特性，例えばセキュリティに関するテストケースも生成されていない. そこで，ユーザーストーリーの記述の有無に関わらず，テストタイプを明示的に指示することで，テストケースが生成されるか否かを検証した.

(4) ペルソナ

生成 AI を用いることのメリットとして，自分以外の視点でテストケースを生成できることが考えられる. つまり，プロンプト指示部において，誰がテストケースを作成しようとしているかを指示することで，指定された人が欲するテストケースが生成されることを期待する. そこで，ペルソナを明示的に指示することにより，

生成されるテストケースに変化があるかを検証した.

次章では, これらの検証を行った結果について述べる.

4. 検証結果

4.1. テスト技法(エラー推測)

3.5 でも述べた通り, 必須のテストケース(表 3)のうち No.1, No.7 などのテストケースが, エラー推測の適用を指示することで追加されるか否かを検証した.

まず, プロンプト指示部の 1 文目を「以下のユーザーストーリー, タスク, 条件からエラー推測の観点を含めた最適なテストケースを作成してください. 」と変更した. しかしながら, この変更では出力結果には変化がなかった.

そこで, エラー推測に関する表現をいくつか変えて試したところ, 「エラー推測の観点を含めた」の部分を「エラーケース, エッジケースを含めた」とすることで, 期待する結果が得られた. さらに, 図 2 に示すように「下記のパターンを含めてください. 」として観点を箇条書きすることで出力が安定したため, これを採用した.

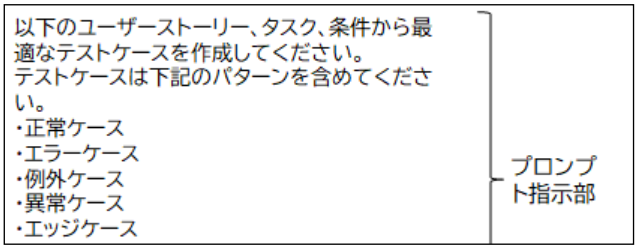

図 2 エラー推測観点の追加

その結果, 出力されたテストケースには表 3 の必須テストケースのうち以下に示す 8 個が含まれ, 基本プロンプトでは出力されなかった No.1 と No.7 のテストケースが出力された.

- 入力受け付け機能のテスト … No.1, No.2, No.3
- 三角形の種類判定機能のテスト … No.4, No.5, No.6, No.7
- 結果表示機能のテスト … No.10
- 性能要件に関するテストケース … なし
- 使用性に関するテストケース … なし

これらに加えて, 任意のテストケース(表 4)のうち, No.17 の特殊な値(int 型の最大値など)のテストケースも出力された. 一方, 必須のテストケースのうち, 結果表示機能のうちの1つ(No.9)や性能要件(No.11)に関するテストケースが出力されなくなるという現象も起きた.

4.2. テスト技法(境界値分析)

必須のテストケース(表 3)の No.8 に該当するケースについて, 境界値分析の適用を指示することで, テストケースが追加されるか否かを検証した. 具体的には, 図 2 に示したプロンプト指示部のパターンに, エラー推測観点に加えてさらに「境界値分析」を追加した. その結果, 本検証で期待するようなテストケースは追加されなかった.

4.3. テストタイプ

様々なテストタイプのテストケースの生成を検証するために, 図 2 に示したプロンプト指示部のパターンに, エラー推測観点に加えて「性能テストケース」「使用性テストケース」「セキュリティテストケース」を追加した. その結果, 出力されたテストケースには表 3 の必須テストケースのうち以下に示す11個が含まれ, 基本プロンプトでは出力されなかったインターフェースの使いやすさに関するテストケース No.12, No.13 が追加された.

- 入力受け付け機能のテスト … No.1, No.2, No.3
- 三角形の種類判定機能のテスト … No.4, No.5, No.6, No.7
- 結果表示機能のテスト … No.9, No.10
- 性能要件に関するテストケース … なし
- 使用性に関するテストケース … No.12, No.13

さらに, 性能要件やセキュリティに関するテストケースについては, 任意のテストケース(表 4)のうち No.19, No.21 も出力された.

なお, この検証に使用したプロンプト指示部のパターンに, さらに「境界値分析」を追加して検証したところ, 境界値の観点のテストケースは追加されず, 逆に必須のテストケースのうち No.10 が出力されなくなった.

4.4. ペルソナ

プロンプト指示部の1文目に「あなたは○○です. 」という1文を追加することで, その人の視点ならではのテストケースが生成されるかを検証した. 指定するペルソナとして, スクラム開発のロールであるプロダクトオーナー, 開発者, スクラムマスターと, 品質の検証を行うロールである QA の計 4 つで検証した. これらのペルソナの違いにより, 出力されるテストケースにも違いが出ることを期待した. 例えば, プロダクトオーナーは受け入れテストの視点, 開発者は単体テスト寄りの視点といった違いが考えられる.

しかしながら, 結果としてはペルソナを指定することで結果に大きな違いはなく, ペルソナ特有のテストケースは出力されなかった.

5. 考察

5.1. 検証結果に対する考察

今回実施した主要なプロンプトパターンの必須のテストケース(表 3)に対するカバレッジを表 5 に示す.

表 5 必須のテストケースに対するカバレッジ比較

プロンプトのパターン	カバレッジ
基本ケース	62%
エラー推測	62%
エラー推測＋境界値分析	62%
エラー推測＋テストタイプ	85%
エラー推測＋テストタイプ＋境界値分析	54%

検証結果を踏まえて, テスト技法やテストタイプについては, それらを明示的に指示することで, その観点のテストケースを追加できると考えられる. 特に, エラー推測とセキュリティテストは, ユーザーストーリーに仕様が明記されていなくても, テストに関する一般的な知識を持っていれば考えられるテストケースが出力された. ただし, エラー推測に関しては, この技法の名称では理解されず,「エラーケース」や「エッジケース」といった用語を使わないと理解されない点には注意が必要である. 特に「エラー」「異常」「例外」といった用語の定義は所属する組織のルールによって変わることもあるため, すべて指定したほうがエラーケースの出力が安定すると考えられる.

一方, 代表的なテスト技法である境界値分析については正しく適用されなかった. ただ, 今回の検証では単純な入力値の境界値ではなく, 2 つの辺の長さの和と他の 1 つの辺の長さの関係における境界値という高度な適用が必要であったため, 期待する結果が得られなかったと考えられる. 3.5 でも述べた通り, このテストケースはエラー推測として導出できる可能性もあるが, その検証でも生成されなかったことから, エラー推測を活用する能力は未成熟と言える.

また, ペルソナを指定することで様々な視点のテストケースが生成されることを期待したが, 特に変化はなかった. 今回の仕様がペルソナに影響を受けるものではなかったためとも考えられる. 例えばシステムテストと受け入れテストは, ほぼ同じテスト対象をそれぞれ開発者の視点と利用者の視点でテストすることになるため, それらのテストレベルでは違いが出ることを期待したい.

5.2. 検証目的以外の考察

検証の目的とは別に, 検証中に気づいた点や気になった点を挙げる.

今回は 3.1 にも示した通り再学習不可として検証したものの, 同じプロンプトの入力に対して異なる結果が返されることがたびたびあった. また, プロンプトのちょっとした文言の違いだけで結果が変わることもあり, 安定的な結果を求めるのは難しいことが分かった. とはいえ全く異なる情報が出力されることは少なく, テスト設計における現状の生成 AI のレベルを示すことができたといえる.

また, テスト設計担当者の固定観念にとらわれず, 新たな気付きを得られるようなテストケースが作成されることを生成 AI には期待していたが, 驚くような結果はなかった. テストの知識や経験がある人であれば同等のテストケースを作成することは可能であり, それ以上の品質を期待するアドバイザーとして補完するような使い方は現状では難しいと考えられる.

今回は, テストケースの内容のみを検証結果として取り上げたが, 指定した出力項目(テスト観点, 入力値, 期待値)について, 情報の不足, 書き方の不統一などの粗さも見られた.

最後に, テスト設計の課題の一つであった作業時間については, 定型的にプロンプトを実行するのであればテストケースの検討や最適なケースの取捨選択する思考の過程は省略でき効率的となるが, その結果を人が確認する作業に多くの時間を要する. 今回はマイヤーズの三角形問題という回答例が既に分かっているものを題材としたが, 実際の開発においては正解の分からない仕様に対するテストケースを作成しなければならないため, 出力結果の確認にはより多くの時間を要すると考えられる.

6. 結論と今後の課題

6.1. 結論

今回の検証を通して, 生成 AI を使って適切な指示を与えれば, 標準的なテストケースを生成できることが確認できた. 特に, 誤ったテストケースが出力されることはなく, 精度は悪くないと言える. ただ, テストケースの網羅性やピンポイント性という点では完成度が低く, 生成 AI の出力結果をそのまま正式なテストケースとして採用することは難しいことが分かった. そのため, 1 回で完全なテストケースを作成しようとするのではなく, 今回検証したように, 観点を追加しながらテストケースを完成させていく手順が推奨されることが分かった. ただし, 観点の追加が必要か否かについては, テスト設計について十分な知識と経験を持ったエキスパートが判断する必要があると考えられる. 併せて, 出

力項目の欠損や具体的な値の不足といった場合もあるので，出力結果に対するレビューは必要である．

生成 AI を使ったテスト設計が有効な場面としては，プロンプトにて指示する観点を事前に作成しておく必要があるという前提はつくが，ハイスキルのテスト設計担当者がチームにいない場合や完全なテストケースを必要としない PoC などが考えられる．さらに，生成 AI を特定領域にファインチューニングすることができれば，用語のブレを調整したり，業種や特定領域に特有のテストケースを出力できたりすることが期待できる．

6.2. 今後の課題

今回は，マイヤーズの三角形問題をユーザーストーリーに変換したものを用いて検証したが，現実のソフトウェア開発で使われるユーザーストーリーを用いて検証することで，生成 AI を用いたテスト設計の実用化の可能性を探るとともに，生成 AI が理解できる理想的なユーザーストーリーの書き方を定義することにもつながると考える．

今回の検証観点からは，生成 AI が得意とするテスト設計の観点までは判別できなかった．それが分かれば，生成 AI が有効な範囲と人が実施すべき範囲を定義できるため，より効率的なテスト設計プロセスを構築できると考えられる．

今回の検証では大きな違いが出なかったペルソナについては，それぞれのペルソナに特有のテストケースがあるような状況で改めて検証を行い，適切な指示の方法を探りたい．それにより，多角的な視点でのテストが可能になると考えられ，テストの抜け漏れ防止に寄与することが期待される．

参考文献

[1] Daniel Zimmermann , Anne Koziolek , GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver , 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)

[2] 坂本一憲，大規模言語モデルを活用したソフトウェア開発の生産性向上に向けて，スマートエスイーコンソーシアム主催 DX 交流フォーラム

[3] 堀川 尚人，テストプロセスにおける「ChatGPT」の活用の提案，ソフトウェア・シンポジウム 2023 in 仙台

[4] Glenford J. Myers，長尾誠 監訳，松尾正信 訳，ソフトウェア・テストの技法 第二版，近代科学社

付録 1 基本プロンプト

You
#

以下のユーザーストーリー、タスク、条件から最適なテストケースを作成してください。

テストケースは、テスト観点、入力値、期待値を結果形式に従って出力してください。

ユーザーストーリー:

1. **三角形の種類を判定する:**
 - ユーザーはプログラムに三角形の辺の長さを提供し、それが不等辺三角形、二等辺三角形、または正三角形のどれかを判定することを期待しています。

タスク:

1. **入力受け付け機能:**
 - プログラムはユーザーから三角形の辺の長さを受け取る入力機能を提供する。
 - エラーハンドリングを行い、不正な入力があった場合に適切なエラーメッセージを表示する。

2. **三角形の種類判定機能:**
 - プログラムは受け取った辺の長さに基づいて三角形の種類を判定する機能を実装する。
 - 判定はマイヤーズの三角形問題に準拠する。

3. **結果表示機能:**
 - プログラムは判定結果をユーザーに適切に表示する機能を提供する。
 - 判定結果はわかりやすく、理解しやすい形式で表示される。

4. **処理時間短縮機能:**
 - プログラムは処理時間を短縮するために効率的なアルゴリズムを実装する。

条件:

1. **正確性:**
 - 判定はマイヤーズの三角形問題に基づいて行われる必要がある。
 - 結果は正確である必要がある。

2. **迅速性:**
 - 判定プロセスは迅速に行われる必要がある。
 - ユーザーは待ち時間なく結果を受け取ることができる。

3. **使いやすさ:**
 - プログラムは使いやすいインターフェースを提供する必要がある。
 - 入力や結果の表示が分かりやすくなければならない。

4. **拡張性:**
 - プログラムは将来的な拡張を容易にする設計である必要がある。
 - 新しい機能や特性の追加が容易である。

5. **エラーハンドリング:**
 - 入力が不正な場合、適切なエラーメッセージを表示する必要がある。

結果形式：
テストケースNo、テストケース名、テスト観点、入力値、期待値を箇条書きで出力して下さい。
テストケースは分類ごとに出力してください。
テストケースはテストの内容が分かるよう名称を付けてください。
テスト観点は確認したい目的を具体的に述べてください。
入力値と期待値は正確な値を出力してください。
項目は、以下の順序で項目名をつけて出力してください。

テストケースNo：テストケース名称
・テスト観点：
・入力値：
・期待値：

付録 2 基本プロンプトから生成されたテストケース

ChatGPT

入力受け付け機能テストケース:

テストケースNo: 1
・テスト観点: 正常な入力が受け付けられるか
・入力値: 辺の長さが3, 4, 5の場合
・期待値: 入力が正常に受け付けられる

テストケースNo: 2
・テスト観点: 不正な入力が拒否されるか
・入力値: 辺の長さが0, 1, -1の場合
・期待値: エラーメッセージが表示される

三角形の種類判定機能テストケース:

テストケースNo: 3
・テスト観点: 正三角形が正しく判定されるか
・入力値: 辺の長さが3, 3, 3の場合
・期待値: 正三角形と判定される

テストケースNo: 4
・テスト観点: 二等辺三角形が正しく判定されるか
・入力値: 辺の長さが5, 5, 8の場合
・期待値: 二等辺三角形と判定される

テストケースNo: 5
・テスト観点: 不等辺三角形が正しく判定されるか
・入力値: 辺の長さが3, 4, 6の場合
・期待値: 不等辺三角形と判定される

結果表示機能テストケース:

テストケースNo: 6
・テスト観点: 判定結果が適切に表示されるか
・入力値: 正三角形の場合の入力値
・期待値: "入力された三角形は正三角形です"というメッセージが表示される

テストケースNo: 7
・テスト観点: 判定結果が分かりやすいか
・入力値: 二等辺三角形の場合の入力値
・期待値: "入力された三角形は二等辺三角形です"というメッセージが表示される

処理時間短縮機能テストケース:

テストケースNo: 8
・テスト観点: 処理時間が短縮されているか
・入力値: 大きな辺の長さを持つ三角形の場合の入力値
・期待値: 判定結果が迅速に表示される

エラーハンドリングテストケース:

テストケースNo: 9
・テスト観点: 不正な入力に対するエラーメッセージが表示されるか
・入力値: 0を含む辺の長さの場合の入力値
・期待値: "無効な入力です。辺の長さは正の値でなければなりません。"というエラーメッセージが表示される