

認定テスト担当者スペシャリストレベル 生成 AI を使用したテスト (CT-GenAI) シラバス

v1.0 J01

国際ソフトウェアテスト資格委員会



著作権について

著作権表示 © International Software Testing Qualifications Board (以下、ISTQB®)

ISTQB® は、International Software Testing Qualifications Board の登録商標である。

著作権 © 2025、著者の Abbas Ahmad、Gualtiero Bazzana、Alessandro Collino、Olivier Denoo、および Bruno Legeard。

無断転載を禁じる。著者はここに著作権を ISTQB® に譲渡する。著者^著（現在の著作権所有者として）と ISTQB®（将来の著作権所有者として）は、次の使用条件に同意した。

無断転載を禁じる。著者は著作権を ISTQB® に譲渡することをここに記す。著者（現著作権所有者）および ISTQB®（次著作権所有者）は以下の使用条件に同意する。出典が認められた場合、非営利目的での利用の下、このドキュメントから引用できる。認定されたトレーニング会社は、トレーニング資料が ISTQB® 認定のメンバーボードから認定を受けた後にのみ、著者および ISTQB® がシラバスの出典および著作権所有者として認められ、このシラバスをトレーニングコースのベースとして利用し、シラバスに言及したトレーニングコースとして紹介し、提供することができる。

著者および ISTQB® がシラバスの出典および著作権所有者として認められている場合、個人または個人のグループはこのシラバスを記事や書籍の基礎として使用することができる

ISTQB® の書面による承認を得ることなく、このシラバスをその他に使用することは禁止されている

ISTQB® が認めたメンバーボードは、上記の著作権表示をシラバスの翻訳版に再現することを条件として、このシラバスを翻訳することができる

改訂履歴

ISTQB®

バージョン	Date	備考
v1.0	2025/07/25	CT-GenAI v1.0 リリース

JSTQB®

バージョン	Date	備考
v1.0 J01	2026/10/13	CT-GenAI v1.0J01 リリース

目次

内容

0	導入	9
0.1	このシラバスの目的	9
0.2	生成 AI を使用したソフトウェアテスト	9
0.3	テスト担当者のキャリアパス	9
0.4	ビジネス成果	9
0.5	試験可能な学習の目的、ハンズオン目標、および知識の認知レベル	10
0.6	認定	11
0.7	標準の取扱い	11
0.8	詳細レベル	11
0.9	このシラバスの構成	11
1	ソフトウェアテストのための生成 AI の概要 - 100 分	13
1.1	生成 AI の基盤と主要な概念	14
1.1.1	AI スペクトラム: シンボリック AI、古典的機械学習、ディープラーニング、生成 AI	14
1.1.2	生成 AI と LLM の基盤	14
1.1.3	基盤、インストラクションチューニング、推論 LLM	16
1.1.4	マルチモーダル LLM および視覚言語モデル	16
1.2	ソフトウェアテストにおける生成 AI の活用: 基本原則	17
1.2.1	テストタスクの主要な LLM 能力	17
1.2.2	AI チャットボットと LLM を利用したソフトウェアテスト用テストアプリケーション	18
2	効果的なソフトウェアテストのためのプロンプトエンジニアリング - 365 分	19
2.1	効果的なプロンプト開発	21
2.1.1	ソフトウェアテストにおける生成 AI のプロンプトの構造	21
2.1.2	ソフトウェアテストのための主要なプロンプト技法	22
2.1.3	システムプロンプトとユーザープロンプト	23
2.2	ソフトウェアテストタスクにプロンプトエンジニアリング技法を適用する	24
2.2.1	生成 AI によるテスト分析	24
2.2.2	生成 AI によるテスト設計とテスト実装	25

2.2.3	生成 AI による自動リグレッションテスト	27
2.2.4	生成 AI によるテストモニタリングとテストコントロール	28
2.2.5	ソフトウェアテスト用のプロンプト技法の選択	29
2.3	生成 AI の結果を評価し、ソフトウェアテストタスクのプロンプトを改良する	30
2.3.1	テストタスクにおける生成 AI の結果を評価するためのメトリクス	30
2.3.2	プロンプトを評価し、反復的に改良するための技法	31
3	ソフトウェアテストにおける生成 AI のリスク管理 - 160 分	33
3.1	ハルシネーション、推論エラー、バイアス	34
3.1.1	生成 AI におけるハルシネーション、推論エラー、バイアス	34
3.1.2	LLM 出力のハルシネーション、推論エラー、バイアスを特定する	34
3.1.3	ソフトウェアテストタスクにおける生成 AI ハルシネーション、推論エラー、バイアスを軽減する技法	36
3.1.4	LLM の非決定的動作の軽減	36
3.2	ソフトウェアテストにおける生成 AI のデータプライバシーとセキュリティリスク	37
3.2.1	生成 AI の使用に関連するデータプライバシーとセキュリティリスク	37
3.2.2	テストプロセスおよびツール向け生成 AI におけるデータプライバシーと脆弱性	37
3.2.3	生成 AI を用いたテストにおけるデータプライバシー保護とセキュリティ強化のための緩和策	38
3.3	ソフトウェアテストにおける生成 AI のエネルギー消費と環境への影響	39
3.3.1	生成 AI の使用がエネルギー消費と CO2 排出量に及ぼす影響	40
3.4	AI の規制、標準、ベストプラクティスフレームワーク	40
3.4.1	ソフトウェアテストにおける生成 AI に関連する AI 規制、標準、およびフレームワーク	40
4	LLM を利用したソフトウェアテスト用テストインフラストラクチャ - 110 分	42
4.1	LLM を利用したテストインフラストラクチャのアーキテクチャアプローチ	43
4.1.1	LLM を利用したテストインフラストラクチャの主要なアーキテクチャコンポーネントとコンセプト	43
4.1.2	RAG	44
4.1.3	テストプロセスの自動化における LLM 搭載エージェントの役割	45
4.2	ファインチューニングと LLM Ops: ソフトウェアテスト用の生成 AI の運用化	45
4.2.1	テストタスクに対する LLM のファインチューニング	46
4.2.2	ソフトウェアテスト用に LLM をデプロイおよび管理する場合の LLM Ops	47
5	テスト組織への生成 AI の導入と統合 - 80 分	48

5.1	ソフトウェアテストにおける生成 AI の導入のロードマップ	49
5.1.1	シャドーAI のリスク	49
5.1.2	ソフトウェアテストにおける生成 AI 戦略の重要な側面	49
5.1.3	ソフトウェアテストタスク用の LLM/SLM の選択	49
5.1.4	ソフトウェアテストへの生成 AI 導入におけるフェーズ	50
5.2	ソフトウェアテストに生成 AI を導入する際の変更を管理する	51
5.2.1	生成 AI を使用したテストに必要なスキルと知識	51
5.2.2	テストチームで生成 AI 機能を構築する	51
5.2.3	AI 対応のテスト組織におけるテストプロセスの進化	52
6	参考文献	53
7	付録 A - 学習の目的/知識の認知レベル	55
8	付録 B - 学習の目的を含むビジネス成果トレーサビリティマトリックス	57
9	付録 C - リリースノート	64
10	付録 D - 生成 AI 固有の用語	65
11	付録 E - 商標	68

謝辞

このドキュメントは、2025 年 7 月 25 日に ISTQB® 総会によって正式に発表された。

このドキュメントは、International Software Testing Qualifications Board のチーム、Abbas Ahmad（製品オーナー）、Gualtiero Bazzana、Alessandro Collino、Olivier Denoo、および Bruno Legeard（技術マネージャー）によって作成された。

チームは、技術レビューを行ってくれた Anne Kramer 氏、Jedrzey Kwapinski 氏、Samuel Ouko 氏、Ina Schieferdecker 氏、提案と意見を提供してくれたレビューチームとメンバーボードに感謝する。

以下の人物がこのシラバスのレビュー、コメント、投票に参加した。

Albert Laura, Aneta Derkova, Anne Kramer, Arda Ender Torçuk, Baris Sarialioglu, Claire Van Der Meulen, Daniel van der Zwan, Derek Young, Dietmar Gehring, Francisca Cano Ortiz, Gary Mogyorodi, Gergely Ágnes, Horst Pohlmann, Ina Schieferdecker, Ingvar Nordström, Jan Sabak, Jaroslaw Hryszko, Jedrzey Kwapinski, Joanna Kazun, Karol Frühauf, Katalin Balla, Koray Yitmen, Laura Albert, Linda Vreeswijk, Lucjan Stapp, Lukáš Piška, Mario Winter, Marton Siska, Mattijs Kemmink, Matthias Hamburg, Meile Posthuma, Michael Stahl, Márton Siska, Nele Van Asch, Nils Röttger, Nishan Portoyan, Piet de Roo, Piotr Wicherski, Péter Földházi, Péter Sótér, Radoslaw Smilgin, Ralf Pichler, Renzo Cerquozzi, Rik Marselis, Samuel Ouko, Stephanie Ulrich, Stuart Reid, Tal Pe'er, Tamás Gergely, Thomas Letzkus, Wim Decoutere, Zsolt Hargitai, Mark Rutz, Patrick Quilter, Earl Burba, Taz Daughtrey, Judy McKay, Randall Rice, Thomas Adams, Tom Van Ongeval, Sander Mol, Miroslav Renda, Geng Chen, Chai Afeng, Xinghan Li, Klaudia Dussa-Zieger, Arnd Pehl, Florian Fieber, Ray Gillespie, József Kreisz, Dénes Medzihradsky, Ferenc Hamori, Giorgio Pisani, Giancarlo Tomasig, Young jae Choi, Arnika Hryszko, Andrei Brovko, Iliia kulakov, Praveen, Kostas Pashalidis, Ferdinand Gramsamer, A. Berfin Öztaş, Abdullah Gök, Abdurrahman AKIN, Aleya Zuhail İŞIK, Anıl Şahin, Atakan Erdemgil, Aysel Bilici, Azmi YÜKSEL, Bilal Gelik, Bilge Yazıcı, Burak Gel, Burcu ÖZEL, Büşra İlayda Çevik Köken, Can Polat, Canan Ayten Dörtkol (Polat), Cansu Mercan Daldaban, Denizcan Orhun Karaca, Didem Çiçek Bay, Duygu Yalçinkaya, Efe Can Yemez, ELİF CERAV, Emine Tekiner, Emre Aman, Emre Can Akgül, Esra Küçük, Gençay GENÇ, Gül Çalışır Açan, Gül Nihal SİNGİL, Güler GÖK, Gulhanim Anulur, Hakan GÜVEZ, Haktan Bilgehan Dilber, Halil Ibrahim Tasdemir, Hasan Küçükayaz, Hatice Erdoğan, Hatice Kübra Daşdoğan, Hüseyin Sevki ARI, Hyulya Gyuler, İLKNUR NEŞE TUNCAL, Kaan Eminç, Kamil Isik, Koray Danışman, Melisa Canbaz, Merve Guleroglu, Müjde Ceylan, Mustafa Furkan CEYLAN, Nergiz Gençaslan, Nuh Soner Bozkurt, Omer Fatih Poyraz, Onur Ersoy, Özlem Körpe, Özgür Özdemir, Sedat YOLTAY, Selahattin Aliyazıcıoğlu, Sevan Lalikoğlu, Sebastian Malyska, Sevim Öykü Demirel, Tatsiana Beliai, Tayg.

日本語訳については、Japan Software Qualifications testing Board メンバーおよび以下の日本語翻訳ワーキンググループメンバーにより行われた。

日本語翻訳ワーキンググループメンバー：

松木 晋祐（株式会社ベリサーブ／AIQVE ONE 株式会社／JSTQB 技術委員）

大段 智広（DMM.com／JSTQB 技術委員）

佐野 あゆみ（GREE／WFS／JSTQB 技術委員）

永田 哲（JSTQB 技術委員）

福田 圭佑（GREE／WFS／JSTQB 技術委員）

吉澤 聡美（日本電気株式会社／JSTQB 技術委員）

0 導入

0.1 このシラバスの目的

このシラバスは、International Software Testing Qualification Board for Testing with Generative AI (CT-GenAI) 資格の基礎を形成する。ISTQB® では、このシラバスを次のように提供している。

1. メンバーボードに対しては、現地の言語に翻訳し、トレーニングプロバイダーを認定する。メンバーボードは、シラバスを特定の言語ニーズに合わせて調整したり、地域の出版物に合わせて参考資料を変更したりすることができる
2. 認証機関に対して、このシラバスの学習の目的に合わせて現地の言語で試験問題を作成すること
3. トレーニングプロバイダーに対して、コースウェアを作成し、適切な指導方法を決定するため
4. 認定候補者は、認定試験の準備をする（トレーニングコースの一部として、または単独で）
5. 国際的なソフトウェアおよびシステムエンジニアリングコミュニティに対して、ソフトウェアおよびシステムテストの専門職を進歩させるため、また書籍や記事の基礎として

0.2 生成 AI を使用したソフトウェアテスト

Testing with Generative AI 資格は、ソフトウェアテストに生成 AI (GenAI) の使用に携わるすべての人を対象とする。これには、テスト担当者、テストアナリスト、テスト自動化エンジニア、テストマネージャー、ユーザー受け入れテスト担当者、ソフトウェア開発者などの役割の人々が含まれる。この CT-GenAI 資格によるテストは、プロジェクトマネージャー、品質マネージャー、ソフトウェア開発マネージャー、ビジネスアナリスト、IT ディレクター、経営コンサルタントなど、ソフトウェアテストに生成 AI を使用する方法の基本を理解したい人にも適している。

0.3 テスト担当者のキャリアパス

ISTQB® スキームは、キャリアのあらゆる段階で試験専門家に幅広い知識と深さの両方を提供する支援を提供する。生成 AI 認定による ISTQB® 認定テスターテストを達成した人は、コアアドバンストレベル（テストアナリスト、テクニカルテストアナリスト、テストマネージャー、テストエンジニアリング）と、その後のエキスパートレベル（テストコントロールまたはテストプロセスの改善）にも興味があるかもしれない。ISTQB の認定テスタースキームの最新情報については、www.istqb.org を参照すること。

0.4 ビジネス成果

このセクションでは、Testing with Generative AI 認定資格を取得した受験者に期待されるビジネス成果を列挙する。

Testing with Generative AI 認定資格を取得した受験者は、次のことができる。

GenAI-BO1	生成 AI の基本的な概念、機能、制限を理解する
GenAI-BO2	ソフトウェアテスト用に大規模な言語モデルをプロンプトする実践的なスキルを開発する
GenAI-BO3	ソフトウェアテストに生成 AI を使用するリスクと軽減策についての洞察を得る
GenAI-BO4	ソフトウェアテストのための生成 AI ソリューションのアプリケーションについての洞察を得る
GenAI-BO5	組織内でのソフトウェアテストの生成 AI 戦略とロードマップの定義と実装に効果的に貢献する

0.5 試験可能な学習の目的、ハンズオン目標、および知識の認知レベル

学習の目的およびハンズオン目標はビジネス成果を支援し、生成 AI を使用したテストの認定試験作成に使用する。

一般に、このシラバスのすべての内容は、導入、ハンズオン目標、および付録を除き、K1、K2、および K3 レベルで試験可能である。試験問題では、K1 レベルのキーワードの知識、またはすべての K レベルの学習の目的を確認する。

具体的な学習の目的のレベルは各章の冒頭に示しており、次のように分類する。

- K1: 記憶
- K2: 理解
- K3: 適用

学習の目的の詳細と例は付録 A に記載する。

学習の目的で明示的に言及していない場合でも、章見出しの直下にキーワードとして列挙しているすべての用語を覚えておく必要がある。

具体的なハンズオン目標 (HO) は各章の冒頭に示す。各 HO はレベル K2 または K3 の LO にリンクしており、ハンズオン演習を通じて学習を洗練することを目的とする。HO のレベルは次のように分類する。

- H0: これには、演習のライブデモまたは録画したビデオが含まれる場合がある。これは研修生が実施するものではないため、厳密には演習ではない
- H1: ガイド付き演習。研修生は、トレーナーが実行する一連のステップに従う
- H2: ヒント付き演習。研修生には、指定された時間枠内で演習を解決できるように、関連するヒントを含む演習を与える **Certified Tester Testing with Generative AI 認定試験**

Certified Tester Testing with Generative AI 認定試験は、このシラバスに基づいて行われる。試験問題の解答には、このシラバスの複数のセクションに基づく資料の使用が必要になる場合がある。導入、ハンズオン目標、付録を除くシラバスのすべてのセクションが試験可能である。標準、書籍、論文は参考資料として含まれているが、シラバス自体に要約されている内容を超えて、その内容を検討することはできない。

詳細については、「生成 AI を使用した認定テスターテストの試験構造とルール V1.0」ドキュメントを参照すること。

参加要件注: ISTQB® Foundation Level 証明書は、ISTQB® Certified Tester Testing with Generative AI 認定試験を受ける前に取得する必要がある。

0.6 認定

ISTQB® メンバーボードは、コース教材がこのシラバスに従うトレーニングプロバイダーを認定する場合がある。トレーニング提供者は、メンバーボードまたは認定を実施する団体から認定ガイドラインを取得する必要がある。認定コースはこのシラバスに準拠していると認められ、コースの一部として ISTQB® 試験を受けることが許可される。

このシラバスの認定ガイドラインは、ISTQB CT-GenAI 認定ガイドライン文書で定義されている。

0.7 標準の取扱い

品質特性とソフトウェアテストに関連する標準、つまり IEEE や ISO などの基礎レベルのシラバスで参照される標準がある。これらの参照の目的は、フレームワークを提供すること、または読者が必要に応じて追加情報のソースを提供することである。なお、シラバスは標準文書を参考に行っている。標準文書は試験を目的としたものではない。標準の詳細については、第 6 章を参照すること。

0.8 詳細レベル

このシラバスの詳細レベルにより、国際的に一貫したコースと試験が可能になる。この目標を達成するために、シラバスは次の内容で構成される。

- ISTQB® Certified Tester Testing with Generative AI 認定の意図を説明する一般的な指導目標
- 学生が覚えていなければならない用語のリスト
- 各知識領域の学習の目的。達成すべき認知学習の成果を説明
- 主要な概念の説明（一般に認められた文献や標準などの情報源への参照を含む）
- 学習を支援するために推奨される実践の各ハンズオン目標の説明

シラバスの内容は、生成 AI によるテストの知識領域全体を説明するものではない。これは、ISTQB® Certified Tester Testing with Generative AI トレーニングコースでカバーされる詳細レベルを反映している。テストに生成 AI を使用する場合、すべてのソフトウェアプロジェクトに適用できるテストの概念と技法に焦点を当てる。

シラバスでは、ISTQB®用語集に従って、ソフトウェアテストおよび品質保証で使用する用語の用語（つまり、名前と意味）が使用される。

0.9 このシラバスの構成

試験対象となる 5 つの章がある。各章の最上位の見出しは、その章の時間を指定する。章より下位レベルではタイミングは提供されない。認定トレーニングコースの場合、シラバスでは少なくとも 13.6 時間の指導が必要で、次のように 5 つの章に分かれている。

- 第 1 章: 100 分 ソフトウェアテストのための生成 AI の概要

- テスト担当者は、トークン化やマルチモーダル機能など、大規模言語モデル (LLM) の基本を学習する
- テスト担当者は、ソフトウェアテストにおける生成 AI (GenAI) のアプリケーションを調査し、AI チャットボットと LLM を利用したテストツールを区別し、トークン化、コンテキストウィンドウ、マルチモーダルプロンプトを実験する
- 第 2 章: 365 分 効果的なソフトウェアテストのためのプロンプトエンジニアリング
 - テスト担当者は、ソフトウェアテストで生成 AI の効果的で構造化されたプロンプトを作成する方法を学ぶ
 - テスト担当者は、ソフトウェアテストタスクのプロンプトエンジニアリング技法に関する実践的な経験を積み、それを適用する
- 第 3 章: 160 分 ソフトウェアテストにおける生成 AI のリスク管理
 - テスト担当者は、生成 AI を使用してテストするときに、ハルシネーション、推論エラー、バイアスを特定して軽減する方法を学ぶ
 - テスト担当者は、ソフトウェアテストで生成 AI のデータプライバシーとセキュリティの問題に対処する方法を学ぶ
 - テスト担当者はソフトウェアテストで生成 AI のエネルギー消費と環境への影響を学習する
 - テスト担当者は、ソフトウェアテストで生成 AI を倫理的、透明性、安全に使用するための AI 規制、標準、ベストプラクティスを学ぶ
- 第 4 章: 110 分 LLM 搭載ソフトウェアテスト用テストインフラストラクチャ
 - テスト担当者は、検索拡張生成や生成 AI エージェントなどの生成 AI アーキテクチャを調査する
 - テスト担当者は、ソフトウェアテストタスク用に LLM をファインチューニングするプロセスを学習する
 - テスト担当者は、ソフトウェアテストで LLM を展開および管理するための大規模言語モデル運用 (LLMOps) の概念を学習する
- 第 5 章: 80 分 テスト組織への生成 AI の導入と統合
 - テスト担当者は、生成 AI をテストプロセスに統合するための構造化されたロードマップを学習する
 - テスト担当者は、生成 AI をテストプロセスに統合するための組織変革を学ぶ

1 ソフトウェアテストのための生成 AI の概要 - 100 分

キーワード

なし

生成 AI 固有のキーワード

AI チャットボット、コンテキストウィンドウ、ディープラーニング、エンベディング、特徴量、基盤 LLM、生成 AI、生成事前トレーニングトランスフォーマー、インストラクションチューニング LLM、大規模言語モデル、機械学習、マルチモーダルモデル、推論 LLM、シンボリック AI、トークン化、トランスフォーマー

第 1 章の学習の目的とハンズオン目標:

1.1 生成 AI の基盤と主要な概念

- | | | |
|-------------|--------|---|
| GenAI-1.1.1 | (K1) | さまざまなタイプの AI を想起する:シンボリック AI、古典的な機械学習、ディープラーニング、生成 AI。 |
| GenAI-1.1.2 | (K2) | 生成 AI と大規模言語モデルの基本を説明する。 |
| HO-1.1.2 | (H1) | ソフトウェアテストタスクに LLM を使用する場合のトークン化とトークン数の評価を練習する。 |
| GenAI-1.1.3 | (K2) | 基盤大規模言語モデル（ファウンデーションモデル）、インストラクションチューニングモデルと推論 LLM を区別する。 |
| GenAI-1.1.4 | (K2) | マルチモーダル LLM と視覚言語モデルの基本原則を要約する。 |
| HO-1.1.4 | (H1) | ソフトウェアテストタスクのテキスト入力と画像入力の両方を使用して、マルチモーダル LLM のプロンプトを作成して実行する。 |

1.2 ソフトウェアテストにおける生成 AI の活用: 基本原則

- | | | |
|-------------|------|-------------------------------------|
| GenAI-1.2.1 | (K2) | テストタスクの主要な LLM 能力の例を示す。 |
| GenAI-1.2.2 | (K2) | ソフトウェアテストに生成 AI を使用する場合の対話モデルを比較する。 |

1.1 生成 AI の基盤と主要な概念

生成人工知能（GenAI）は、事前にトレーニングされた大規模なモデルを使用して、テキスト、画像、コードなどの人間のような出力を生成する人工知能の分野である。大規模言語モデル（LLM）は、大規模なテキストデータセットで事前トレーニングされた生成 AI モデルで、コンテキストを判断し、ユーザープロンプトに従って関連する応答を生成できるようにする。

主要な概念には、トークン化（効率的な処理のためにテキストを単位に分割する）、コンテキストウィンドウ（関連性を維持するために一度に考慮される情報の量を制限する）、マルチモーダルモデル（リッチなインタラクションのためにテキスト、画像、音声などの複数のデータ型を処理できる）が含まれる。

ソフトウェアテストでは、これらの LLM は、テストプロセス全体にわたって、受け入れ基準のレビューと改善、テストケースまたはテストスクリプトの生成、潜在的な欠陥の特定、欠陥パターンの分析、合成テストデータの生成、ドキュメント生成の支援などのタスクを支援できる。

1.1.1 AI スペクトラム: シンボリック AI、古典的機械学習、ディープラーニング、生成 AI

人工知能（AI）は、さまざまな種類のテクノロジーを含む幅広い分野であり、それぞれが問題を解決する独自の方法を持っている。例えば、シンボリック AI、古典的な機械学習、ディープラーニング、生成 AI などがある（このシラバスの範囲外のテクノロジーも含む）。

- シンボリック AI は、ルールベースのシステムを使用して人間の意思決定を模倣する。基本的に、シンボリック AI は記号と論理ルールを使用して知識を表現する
- 古典的な機械学習は、データの準備、特徴量の選択、モデルのトレーニングを必要とするデータ駆動型のアプローチであり、欠陥の分類やソフトウェアの問題の予測などのタスクに使用できる
- ディープラーニングは、ニューラルネットワークと呼ばれる機械学習構造を使用して、データから特徴量を自動的に学習する。深層学習モデルは、ユーザーが手動で特徴量を定義することなく、画像、ビデオ、オーディオ、テキストなどの非常に大規模で複雑なデータセット内のパターンを見つけることができるが、実際には、データの注釈、モデルの調整、結果の検証などのタスクに人間の関与が必要になる場合がある
- 生成 AI は、深層学習技術を使用して、トレーニングデータからパターンを学習および模倣することで、新しいコンテンツ（テキスト、画像、コード）を作成する。LLM などのモデルは、トレーニングの範囲内でテキストを生成し、コードを記述し、推論や問題解決をシミュレートできる

要約すると、AI の分野はいくつかの方向に進化しており、それぞれに異なる長所と限界がある。ソフトウェアテストに生成 AI を使用する主な利点は、追加のトレーニングフェーズを必要とせずにテストタスクに直接適用できる事前トレーニング済みモデルを使用できることであるが、これにはいくつかのリスクが伴う（3.1 節を参照）。

1.1.2 生成 AI と LLM の基盤

LLM は、事前にトレーニングされた生成トランスフォーマー深層学習モデルに基づいて、書籍、記事、Web サイトなどの非常に大規模なデータセットでトレーニングされている。小規模言語モデル（SLM）

は、大規模言語モデルと比較してパラメータが少ないコンパクトなモデルであり、軽量で焦点を絞った生成 AI ソリューションを提供するように設計されている。

LLM は、言語のニュアンスを処理できるだけでなく、一貫したコンテンツを生成できる。LLM のコンテンツの処理と生成に役立つ 2 つの重要な概念は、トークン化とエンベディングである。トークン化とエンベディングは、言語をモデルが効果的に処理できる数値形式に変換する。

- 言語モデルにおけるトークン化は、テキストをトークンと呼ばれる小さな単位に分割するプロセスである。トークンは、文字単位の小さな場合もあれば、サブワードや単語単位の大きな場合もある。LLM が文を処理するとき、全体のコンテキストを維持しながら各トークンを個別に理解できるように、最初に入力をトークン化する
- エンベディングは、生成 AI モデルによる処理に適した形式で意味論的、構文論的、および文脈上の関係をエンコードするトークンの数値表現である。各トークンは高次元空間でベクトルに変換され、その意味と使用法に関する微妙な情報が捉えられる。同様の意味または文脈上の役割を持つトークンには、この空間内で密接に配置されたエンベディングがある。この近接性により、LLM は単語の関係を理解し、文脈を保持し、一貫性があり文脈上適切な応答を生成することができる

LLM は、トランスフォーマーモデルとして知られるニューラルネットワークアーキテクチャを利用する。トランスフォーマーモデルは、広範なテキストシーケンスのコンテキストを処理し、トークンが相互にどのように関係するかを学習することにより、言語タスクに優れている。推論中、LLM はシーケンス内の次のトークンを予測し、これらの学習された関係を活用して、一貫性があり文脈上適切なテキストを生成する。トランスフォーマーモデルを使用すると、トレーニングデータとプロンプトに基づいて、統計的に妥当な新しいテキストを生成できる。しかし、もっともらしいことが必ずしも正しいとは限らない。

LLM は、主に推論メカニズムとハイパーパラメータ設定の確率的な性質により、非決定的な動作を示す。この固有のランダム性により、同じ入力複数回提供された場合でも、出力が変動する可能性がある。

LLM の領域におけるコンテキストウィンドウとは、モデルが応答を生成するときに考慮できる、トークンで測定された先行テキストの量を指す。コンテキストウィンドウが大きくなると、長いパッセージにわたってモデルが一貫性を維持できる（例えば大規模なテストログを分析する場合など）。ただし、コンテキストウィンドウ内のトークンの数が増えると、モデルを効果的に実行するために必要な計算の複雑さと処理時間も増加する。

ハンズオン目標 HO-1.1.2 (H1) : トークン化とトークン数の評価の実践

この実践的なアクティビティは、受講者が LLM を使用する際のトークン化とその影響について実践的に理解できるように設計されている。この演習は 2 つの主要な部分に分かれている。

- トークン化: トークナイザーを使用して、サンプルテキストを個々のトークンに分割する。出力を調べて、単語、句読点、フレーズがどのように表現されているかを確認し、トークン化のパターンやニュアンスを特定する

- トークン数の評価: さまざまな入力テキストから生成されたトークンの数を測定する。特にモデルのコンテキストウィンドウの制限と効率の考慮事項に関連して、トークン数がモデルのパフォーマンスにどのような影響を与えるかを分析する

この演習が終了するまでに、受講者は、さまざまなテキスト構造と入力長が LLM との対話にどのような影響を与えるかをよりよく予測できるようになる。

1.1.3 基盤、インストラクションチューニング、推論 LLM

大規模言語モデルは、段階的な専門化トレーニングを通じて、幅広いタスクへの有効性を高める。これらの段階では、基盤 LLM、インストラクションチューニング LLM、推論 LLM の 3 つの主要カテゴリが生成される。

- **基盤 LLM:** これらは、テキスト、コード、画像、その他の形式で構成される膨大で多様なデータセットでトレーニングされた汎用モデルである。広範な事前トレーニングにより、自然言語処理、コンピュータービジョン、音声認識などの分野にわたるさまざまなタスクを支援できるようになる。強力かつ柔軟である一方で、基盤モデルは通常、特定のタスク要件を満たすためにさらに適応する必要がある
- **インストラクションチューニング LLM:** 基盤モデルから派生したインストラクションチューニング LLM は、プロンプトと予想される応答を組み合わせるデータセットを使用してファインチューニングされる。この段階では、人間のインストラクション(指示)との整合性が強化され、現実世界のアプリケーションでの使いやすさが向上する。チューニングプロセスには、タスクの順守、指示の追従、応答の一貫性の最適化が含まれており、これにより、ユーザーの意図を効果的に解釈して実行するモデルの能力が向上する
- **推論 LLM:** 推論モデルは、論理的推論、複数ステップの問題解決、思考連鎖推論などの構造化された認知能力を強調することにより、命令調整モデルを拡張する。これらのモデルは、文脈の理解、中間の推論ステップ、複雑な情報の合成を必要とする、慎重に選択されたタスクでさらにトレーニングまたはファインチューニングされる。その結果、技術的な領域を含む、認知負荷の高いタスクに適している

ソフトウェアテスト用の生成 AI アプリケーションのコンテキストでは、命令チューニング（非推論とも呼ばれる）と推論 LLM の両方が利用される。選択は、対象とする特定のテストタスクの複雑さと推論の要求によって異なる。

1.1.4 マルチモーダル LLM および視覚言語モデル

マルチモーダル LLM は、従来のトランスフォーマーモデルを拡張して、テキスト、画像、サウンド、ビデオなどの複数のデータ形式を処理する。これらのモデルは、大規模で多様なデータセットでトレーニングされており、異なる種類のデータ間の関係を学習できる。さまざまな形式を処理するために、トークン化は各データタイプに適合される。例えば、画像は、トランスフォーマーモデルで処理される前に、視覚言語モデルを使用してエンベディングに変換される。

マルチモーダル LLM のサブセットである視覚言語モデルは、特に視覚情報とテキスト情報を統合して、画像キャプション、視覚的な質問応答、テキスト入力と視覚入力間の一貫性の分析などのタスクを実行する。

ソフトウェアテストでは、マルチモーダル LLM、特に視覚言語モデルで強化された LLM が重要な機会を提供する。スクリーンショットや GUI ワイヤフレームなどのアプリケーションの視覚要素を、欠陥レポートやユーザーストーリーなどの関連するテキスト説明とともに分析できる。この機能により、テスト担当者は、期待結果とスクリーンショット上の実際の視覚要素との間の相違点を特定できる。さらに、視覚言語モデルで強化された LLM は、テキストデータと視覚的手がかりの両方を組み込んだ豊富で現実的なテストケースを生成できるため、全体的なカバレッジが向上する。

ハンズオン目標 HO-1.1.4 (H1) : マルチモーダル LLM モデルを使用して、テストタスクに対処する指定されたプロンプトを確認して実行する

この演習では、テキストと画像の両方の入力を使用して、マルチモーダル LLM の指定されたプロンプトを確認して実行し、次の 2 つのステップでテストタスクを解決する。

- 入力を確認する: プロンプトと入力データ (テキストと画像) を確認する
- プロンプトを実行して結果を確認する。マルチモーダル LLM を使用して画像とテキストの両方を入力し、LLM の応答を確認する

この演習では、ソフトウェアテストのユースケースでテキストと画像の両方の入力を伴うタスクにマルチモーダル LLM を使用する方法を示す。これには、関連する利点と潜在的な課題の認識も含まれる。

1.2 ソフトウェアテストにおける生成 AI の活用: 基本原則

生成 AI は、さまざまなテスト活動に革新的な機能を提供する。LLM は、自然言語とコードの処理、一貫したテキストとコードの生成、質問への回答、情報の要約、言語の翻訳、マルチモーダルなコンテキストでの画像の分析に優れている。

あらゆる役割のテスト専門家は、クエリに即座に応答する生成 AI チャットボットを使用する方法と、テストツールに統合された LLM を利用したアプリケーションを使用する方法の 2 つの補完的な方法で生成 AI を活用できる。

1.2.1 テストタスクの主要な LLM 能力

LLM は、要件、仕様、スクリーンショット、コード、テストケース、および欠陥レポートを解釈できるため、テストプロセス全体に必要な情報を理解し明確にし、テストウェアの要素を生成するためのツールとなる。以下は、ソフトウェアテストに関連する主要な LLM 能力の一部である。

- 要件の分析と改善: LLM は、曖昧さ、矛盾、欠落している情報を特定することにより、要件やテストベースのその他の要素の分析に役立つ。利害関係者との議論中に要件を明確にするのに役立つ有意義な質問を生成できる
- テストケース作成の支援: LLM は、テストケースの生成を支援し、システム要件、ユーザーストーリー、またはその他のテストベースの要素に基づいてテスト目的を提案する
- テストオラクルの生成: LLM は、期待結果の生成に役立つ

- テストデータの生成:LLM は、データセットを生成し、境界値を設定し、テストデータのさまざまな組み合わせを作成できる
- テスト自動化支援:LLM は、テストケースの記述からテストスクリプトを生成し、変更を提案し、適切なテスト設計技法を特定することで、既存のテストスクリプトを改善するのに役立つ
- テスト結果の分析:LLM は、概要を作成し、重要度と優先度に基づいて異常を分類することで、テスト結果の分析に役立つ
- テストウェアの作成:LLM は、テスト計画、テストレポート、欠陥レポートなどのさまざまなドキュメントの作成を支援し、プロジェクトの進行に合わせてそれらを最新の状態に保つことができる

これらの機能は、LLM がテストプロセス全体を通じてソフトウェアテストのさまざまな側面にどのような影響を与えるかを示している。

1.2.2 AI チャットボットと LLM を利用したソフトウェアテスト用テストアプリケーション

AI チャットボットと LLM を利用したテストアプリケーションは、機能、柔軟性、統合アプローチが異なるが、どちらもテスト担当者を支援できる。

AI チャットボットは、テスト担当者が LLM と直接コミュニケーションを取れるようにする、ユーザーフレンドリーな会話型インターフェースを提供する。この自然言語による対話により、テスト担当者は質問、コマンド、またはプロンプトを入力し、状況に応じた応答を即座に受け取ることができる。プロンプトチェーンなどの技術を通じて、テスト担当者は繰り返し出力を改良することができ、チャットボットは、テストの知識と実践への迅速なアクセスを提供することで、ルーチンタスク、探索的テスト、さらには新しいテスト担当者の新人研修にも特に効果的になる。

これらの AI チャットボットは、迅速なフィードバック、テスト概念の明確化、要件と潜在的なテストケースの動的な探索を必要とするシナリオで特に有益である。直観的なインターフェースにより、技術者以外の関係者でもアクセスしやすくなり、潜在的なユーザーベースが広がり、より幅広い採用が促進される。

対照的に、LLM を利用したテストアプリケーションには、API を介した LLM 機能の統合が含まれており、明確に定義され、多くの場合自動化されたテストタスクを実行する。これらのアプリケーションは優れたカスタマイズ性とスケーラビリティを提供し、組織やツールベンダーが生成 AI を既存のテストフレームワークに組み込むことができる。これにより、テストケースの生成、欠陥分析、テストデータの合成など、反復的なタスクや複雑なタスクの自動化が可能になる。より高度な実装では、組織は特定のテストの役割を実行するために特別に設計された AI エージェントを作成できる（第 4 章を参照）。

テスト担当者が LLM とどのように対話するかチャットボット経由か、LLM を利用した統合アプリケーションを通じてかにかかわらず、テストにおける生成 AI の実装を成功させるには、強力なプロンプトエンジニアリングが必要である（第 2 章を参照）。LLM が生成する出力が正確で関連性があり、テストの目的と一致していることを確認するには、慎重に設計されたプロンプトと明確で具体的な指示が不可欠である。この実践は、生成 AI から得られる価値を最大化するのに役立ち、幅広いテスト活動に対する一貫した信頼性の高い支援を保証する。

2 効果的なソフトウェアテストのためのプロンプトエンジニアリング - 365 分

キーワード

受け入れ基準、テストスクリプト、テストケース、テスト条件、テストデータ、テスト設計、テストレポート

生成 AI 固有のキーワード

Few-Shot プロンプト、メタプロンプト、自然言語処理、ワンショットプロンプト、プロンプト、プロンプトチェーン、プロンプトエンジニアリング、システムプロンプト、ユーザープロンプト、ゼロショットプロンプト

第 2 章の学習の目的とハンズオン目標:

2.1 効果的なプロンプト開発

GenAI-2.1.1	(K2)	ソフトウェアテスト用の生成 AI で使用されるプロンプトの構造の例を挙げる。
HO-2.1.1	(H0)	ソフトウェアテストタスクのいくつかの指定されたプロンプトを観察し、それぞれの役割、コンテキスト、指示、入力データ、制約、出力形式のコンポーネントを特定する。
GenAI-2.1.2	(K2)	ソフトウェアテストのためのコアプロンプト技法を区別する。
HO-2.1.2a	(H0)	ソフトウェアテストタスクに適用されるプロンプトチェーン、Few-Shot プロンプト、およびメタプロンプトのデモンストレーションを観察する。
HO-2.1.2b	(H1)	指定された例でどのプロンプトエンジニアリング技法が使用されているかを特定する。
GenAI-2.1.3	(K2)	システムプロンプトとユーザープロンプトを区別する。

2.2 ソフトウェアテストタスクへのプロンプトエンジニアリング技法の適用

GenAI-2.2.1	(K3)	生成 AI をテスト分析タスクに適用する。
HO-2.2.1a	(H2)	GUI ワイヤフレームに基づいてユーザーストーリーの受け入れ基準を生成するマルチモーダルプロンプトを練習する。
HO-2.2.1b	(H2)	プロンプトチェーンと人間による検証を実践して、特定のユーザーストーリーを段階的に分析し、受け入れ基準を調整する。
GenAI-2.2.2	(K3)	生成 AI をテスト設計およびテスト実装タスクに適用する。
HO-2.2.2a	(H2)	プロンプトチェーン、構造化プロンプト、メタプロンプトを使用した生成 AI によるユーザーストーリーからの機能テストケースの生成を実践する。
HO-2.2.2b	(H2)	Few-Shot プロンプト技法を使用して、ユーザーストーリーから Gherkin スタイルのテスト条件とテストケースを生成する。
HO-2.2.2c	(H2)	プロンプトチェーンを使用して、特定の優先順位と依存関係を考慮して、特定のテストスイート内のテストケースに優先順位を付ける。

GenAI-2.2.3	(K3)	自動リグレーションテストに生成 AI を適用する。
HO-2.2.3a	(H2)	キーワード駆動型のテストスクリプトを作成および管理するための、 Few-Shot プロンプトを練習する。
HO-2.2.3b	(H2)	テストレポート分析のための構造化されたプロンプトエンジニアリングを実践する。
GenAI-2.2.4	(K3)	生成 AI をテスト制御および監視タスクに適用する。
HO-2.2.4	(H0)	テストデータから生成 AI が作成したテストモニタリングメトリクスを観察する。
GenAI-2.2.5	(K3)	特定のコンテキストとテストタスクに適切なプロンプト技法を選択して適用する。
HO-2.2.5	(H1)	特定のテストタスクに対して、状況に応じたプロンプト技法を選択して適用する。

2.3 生成 AI の結果を評価し、ソフトウェアテストタスクのプロンプトを調整する

GenAI-2.3.1	(K2)	テストタスクにおける生成 AI の結果を評価するためのメトリクスを理解する。
HO-2.3.1	(H0)	テストタスクでの生成 AI の結果を評価するためにメトリクスをどのように使用できるかを観察する。
GenAI-2.3.2	(K2)	プロンプトを評価し、反復的に調整するための技法の例を示す
HO-2.3.2	(H1)	特定のテストタスクのプロンプトを評価して最適化する、

2.1 効果的なプロンプト開発

効果的なプロンプト設計により、生成 AI ツールはソフトウェアテストタスクを正確かつ効率的に実行し、テスト担当者は LLM から有益な結果を得ることができる。構造化プロンプトにはさまざまなコンポーネントが含まれる (2.1.1 項を参照)。これらの各コンポーネントは、LLM に要件と期待を効果的に伝えるプロンプトの明瞭さと正確さに貢献する。

さまざまなプロンプトエンジニアリング技法により、ソフトウェアテストにおけるプロンプトの有効性が向上する。プロンプトチェーン、Few-Shot プロンプト、メタプロンプトなどの技法は、複雑なテストの課題に対処するのに役立つ (2.1.2 項を参照)。

構造化プロンプト (2.2.1 節を参照) とコアプロンプト技法を組み合わせることで、ソフトウェアテストタスクで LLM をクエリするときに良好な結果を達成することを目的としている (2.1.3 項を参照)。

2.1.1 ソフトウェアテストにおける生成 AI のプロンプトの構造

ソフトウェアテスト用の構造化プロンプトには通常、次の 6 つのコンポーネントが含まれる。

- 役割: 生成 AI モデルが応答を生成するときに取るべき視点またはペルソナを定義する。役割を指定すると、LLM がその責任を決定し、テスト担当者、テストマネージャー、テスト自動化エンジニアとしての役割など、適切なトーンやアプローチを採用するのに役立つ
- コンテキスト: 生成 AI モデルがテスト条件を決定するために必要な背景情報を提供する。これには、テストオブジェクト、テストする特定の機能、および関連するコンテキスト情報に関する詳細が含まれる
- 指示: 生成 AI に与える指令であり、実行すべき具体的なタスクの概要を示す。明確で命令的かつ簡潔な指示には、タスクの説明とタスクに関連する要件が含まれる
- 入力データ: ユーザーストーリー、受け入れ基準、スクリーンショット、コード、既存のテストケース、出力例など、タスクを実行するために必要な情報が含まれる。詳細で構造化された入力データを提供することにより、LLM はより正確でコンテキストを認識した結果を生成することができる
- 制約: LLM が従うべき制限または特別な考慮事項の概要を示す。制約は、入力データに命令を適用する方法を指定するのに役立つ
- 出力形式: 応答の予期される形式、構造、または特性を示す。これらのインジケータは、LLM の出力を形成するのに役立つ

これらのコンポーネントは、プロンプトの基本構造を形成する。この構造は、実行するタスクと使用する LLM に応じて、プロンプト技法 (2.1.2 項を参照) の実装と組み合わせる必要がある。

ハンズオン目標 HO-2.1.1 (H0) : プロンプトコンポーネントを観察および分析する

デモンストレーションでは、AI チャットボット上でいくつかの構造化されたプロンプトが実験し、それぞれが特定のソフトウェアテストタスクに合わせて調整する。これらのプロンプトは、役割、コンテキスト、指示、入力データ、制約、出力形式という 6 つの主要なコンポーネントで構成される構造化形式に従う。このデモンストレーションは、これらの構造化されたプロンプトの観察と分析を容易

にすることを目的としており、ソフトウェアテストタスクに使用される LLM に正確で関連性のある実用的な洞察を提供することに、各コンポーネントがどのように貢献するかを強調する。

2.1.2 ソフトウェアテストのための主要なプロンプト技法

近年、さまざまな生成 AI の使用例に対して、多くの LLM プロンプト技法が提案されている (Schulhoff 2024)。これらの中で、プロンプトチェーン、Few-Shot プロンプト、メタプロンプトという 3 つのコアプロンプト技法が、上記の 6 コンポーネントプロンプト構造 (2.1.1 項を参照) と組み合わせて生成 AI を使用したテストタスクによく利用される。

- プロンプトチェーンには、タスクを一連の中間ステップ (複数のプロンプト) に分割することが含まれる。各ステップの結果は、次のステップに進む前に手動または自動でチェックされ、調整される。このアプローチにより、各応答が次のプロンプトに通知されるため、精度が向上する。プロンプトチェーンは、タスクが複雑で、サブタスクへの分解と中間 LLM 出力の体系的なチェックが必要なテストプロセスで特に役立つ。また、テストプロセスでの動的な対話も可能になる
- Few-Shot プロンプトには、プロンプト内で LLM に例を提供することが含まれる。ゼロショットプロンプト (例なし) はモデルの既存の知識に依存して応答を生成するが、ワンショットプロンプトは、特定の入力に対して望ましい結果を示す 1 つの例を提供する。Few-Shot プロンプトには、モデルの望ましい応答動作をさらに統合するために複数の例 (数例) が含まれている

この技法は、明確な参照を提供し、結果に一貫性があり期待と一致していることを確認することで、モデルをガイドするのに役立つ。Few-Shot プロンプトは、例で必要な動作を説明できるタスクに特に効果的で、モデルを効果的に一般化し、信頼性の高い結果を生成できる

- メタプロンプトは、AI が自身のプロンプトを生成または改善する能力を活用する。反復的なサイクルにおいて、LLM はプロンプトを生成し、テスト担当者はそれを評価して改善できる。このアプローチでは、最適化されたプロンプトに関する LLM の知識を利用して、プロンプトの品質を最適化する。メタプロンプトは、効果的なプロンプトを設計するために必要な手動の労力を軽減することから、効率とプロンプトの最適化が重要な場合に特に有益である。メタプロンプトのもう 1 つの利点は、テスト担当者が効果的なプロンプトを作成する方法がわからない場合に、LLM と協力してプロンプトを共同作成できることである。これは、テスト担当者と AI がインタラクティブに連携して共通の目標を達成する生成 AI ツールとのペアリングの形式を反映している。このペアリングの概念は、AI ツールと連携する新しい方法を強調し、プロンプトエンジニアリングだけでなく、ペアプログラミングやペアテストの生産性と学習の両方を強化する

これらのプロンプト技法を効果的に組み合わせて使用すると、LLM の結果を改善できる (2.2.5 項を参照)。

ハンズオン目標 HO-2.1.2a (H0) : ソフトウェアテストタスクにおけるプロンプトチェーン、Few-Shot プロンプト、およびメタプロンプトを観察し、議論する

参加者は、AI チャットボットでのプロンプトチェーン、Few-Shot プロンプト、およびメタプロンプトを体験する。それぞれが特定のソフトウェアテストタスクに適用される。このデモンストレーション

は、ソフトウェアテストのコンテキストでこれらのプロンプト技法を調査および議論することを目的としており、各技法が LLM 出力の精度と完全性にどのように貢献するかを強調する。

ハンズオン目標 HO-2.1.2b (H1) : 与えられた例におけるプロンプトエンジニアリング技法を特定する

参加者は、ソフトウェアテストに関連する一連のプロンプトの例を読んで、適用されている中心的なプロンプト技法を特定する。プロンプトチェーン、**Few-Shot** プロンプト、メタプロンプトなどの技法を認識することに重点を置き、その際立った特徴と実際の応用例を強調する。

このアクティビティは、さまざまなプロンプト技法がソフトウェアテストにおける生成 AI の効果的な使用をどのように強化するかについて、参加者の理解を深めることを目的としている。

2.1.3 システムプロンプトとユーザープロンプト

LLM との対話では、システムプロンプトとユーザープロンプトは異なる目的を果たし、それぞれが会話の形成において異なる役割を果たす。システムプロンプトは通常、LLM の全体的な動作をガイドするために開発者またはテスト担当者によって定義され、ほとんどのインターフェースではチャットボットのユーザーは表示または編集できない。

システムプロンプトは、LLM の動作、性格、操作パラメータを定義する事前定義されたコマンドセットとして機能する。動作パラメータによって、LLM の応答方法が決まる。例えば、フォーマルな文体を使用する、回答を簡潔に保つ、ドメイン固有のルールを尊重する、特定の動作を回避するなどである。システムプロンプトは、会話全体のルールを設定する。ロール、コンテキスト、制約などの構造化プロンプトの一部が含まれる場合がある。

システムプロンプトは対話セッション全体を通じて一定に保たれ、LLM がどのように応答するかについての基本的なフレームワークを確立する。例えば、システムプロンプトでは、次のように記述される場合がある。「あなたはプロのソフトウェアテストアシスタントである。常に明確に回答し、フォーマルな言葉を使用し、ISTQB に合わせた実践に焦点を当てること。推測は避け、必要に応じてテスト原則を引用すること。」

一方、ユーザープロンプトは、チャットボットのユーザーからの実際の入力または質問を示す。これは対話ごとに変化し、チャットボットのユーザーが LLM に対処してもらいたい特定の指示、質問、またはタスクを含めることができる。システムプロンプトとは異なり、ユーザープロンプトは直接表示され、各応答の直接のコンテキストを形成する。

例えば、ユーザープロンプトは次のようになる。「ブラックボックステストとホワイトボックステストの主な違いを例とともに挙げなさい。」

典型的な使用法では、会話の開始時にシステムプロンプトを 1 回設定し、その後、対話ごとに連続したユーザープロンプトを送信する。LLM は、不変のシステムプロンプトと現在のユーザープロンプトの両方を考慮して応答を生成する。効果的な実装のために、システムプロンプトは LLM の役割と考えられる制約について明確かつ具体的である必要がある。また、例えば期待される出力に対して、コンテキストや一般的な指示も含まれる場合がある。

ユーザープロンプトは、明示的な指示だけでなく、追加の関連コンテキストや出力形式の指示も含めて、焦点を絞り、適切に構造化されている必要がある。

2.2 ソフトウェアテストタスクにプロンプトエンジニアリング技法を適用する

ソフトウェアテストにプロンプトエンジニアリング技法を適用することで、生成 AI は、テスト分析、テスト設計、テスト自動化、テストケースの優先順位付け、欠陥検出、カバレッジ分析、テストモニタリングとテストコントロールなどのテストタスクを支援できるようになる。プロンプトチェーン、Few-Shot プロンプト、メタプロンプトなどの技法を使用および組み合わせることにより、チームは AI プロンプトを特定のテスト目的に合わせて調整し、出力をより正確で、関連性があり、効果的なものにすることができる。有意義な AI 結果を得るには、高品質の入力が不可欠である。

2.2.1 生成 AI によるテスト分析

生成 AI は、テスト条件の生成と優先順位付け、テストベースの欠陥の特定、カバレッジ分析の提供により、テスト分析タスクを支援できる。入力データには、要件、ユースケースストーリー、技術仕様、GUI ワイヤフレーム、その他の関連情報が含まれる。出力は、優先順位付けされたテスト条件（受け入れ基準など）などの典型的なテスト分析作業成果物で構成される。

生成 AI で支援できる典型的なテスト分析タスクをいくつか示す。

- **テストベース内の潜在的な欠陥を特定する:** 生成 AI は、欠陥につながる可能性のある不整合、曖昧さ、または不完全な情報についてテストベースを分析するのに役立つ。LLM は、同様の要件パターンを比較したり、以前の欠陥レポートからの知識を適用したりすることで、潜在的な異常にフラグを立て、改善を提案できる
- **テストベース（要件/ユースケースストーリーなど）に基づいたテスト条件の生成:** LLM は要件とユースケースストーリーを分析してテスト条件を生成できる。自然言語処理を使用すると、要件の意味を解釈し、測定可能でテスト可能なステートメントに分解できる。これは、要件を特定のテスト条件に変換するのに役立つ
- **リスクレベルに基づいてテスト条件に優先順位を付ける:** 各テスト条件に対するリスクの可能性と失敗のリスク影響に関する情報を利用して、LLM はテスト作業の優先順位付けに役立つ。LLM は、法規制への準拠、ユーザー向け機能（ログイン機能や支払い処理など）、および過去の欠陥データなどの側面を考慮することで、優先レベルを推奨できる
- **カバレッジ分析の支援:** 要件とユースケースストーリーをテスト条件にマッピングすることにより、LLM はカバレッジ分析を実行して、テストベースのすべての側面がカバーされているかどうかを判断できる。これは特に、要件が複雑で、カバレッジの漏れが欠陥の流出につながる恐れがあるプロジェクトで特に役に立つ
- **テスト技法の提案:** 生成 AI は、テスト対象の要件またはユースケースストーリーの種類に基づいて、関連するテスト技法（境界値分析、同値分割など）を提案できる。これは、テスト担当者が特定のテスト条件に対して最も効果的なテスト技法を適用するのに役立つ

完了すべきタスクに対して LLM に提供する入力の品質と関連性は、LLM が生成する出力の正確性と精度に直接影響する。

ハンズオン目標 2.2.1a (H2) :GUI ワイヤフレームに基づいてユーザーストーリーの受け入れ基準を生成する、構造化されたマルチモーダルプロンプトを作成する練習をする

これは、マルチモーダル入力（テキストと画像）を使用して構造化プロンプトを作成する練習である。目標は、ユーザーストーリーと GUI ワイヤフレームから高品質な（つまり、整った、明確で完全）受け入れ基準を生成することである。他のテキスト要素を追加して、入力フィールドの制約やデータ処理に適用するビジネスルールなどのコンテキストを提供できる。

LLM から得られた結果を比較し、テスト分析タスクに対する構造化プロンプトのさまざまな形式（役割、コンテキスト、指示、テキストおよび画像の入力データ、制約、出力形式）の影響を評価する。

この演習では、LLM から正確で関連性の高い結果を取得する際の、プロンプトの構造化の重要性、正確な指示の貢献、テキストと画像の両方のコンテキストデータの重要性について実践的な経験を提供する。

ハンズオン目標 2.2.1b (H2) : 特定のユーザーストーリーを段階的に分析し、受け入れ基準を調整するために、プロンプトチェーンと人間による検証を実践する

これは、最初に曖昧さを特定し、次にテスト容易性を評価し、最後に完全性を評価することによって、特定のユーザーストーリーを分析し、受け入れ基準を洗練するためのプロンプトチェーンを実践するための演習である。この演習では、ステップバイステップのアプローチを奨励し、各ステップで分析を洗練して、テストの目的を達成するために受け入れ基準が適切に形成され、実行可能であることを確認する。各ステップで、LLM が提供する結果を手動で検証し、必要に応じて、出力を調整するか、LLM によるプロンプトチェーンプロセスを通じて修正する。このように、次のステージでは、前のステージのクリーンな結果を使用して、受け入れ基準を改善する別の側面に取り組む。

この演習では、各段階の結果を人間が検証しながら、複雑なタスクをサブタスクに分割する利点を実際に体験する。

2.2.2 生成 AI によるテスト設計とテスト実装

[ISTQB_CTFL_SYL]で説明されているように、テスト設計にはテスト条件の精緻化と改良が含まれ、その後、テストケースやその他のテストウェアに変換される。テストの実装には、テストを実行するために必要なテストウェアの作成または取得が伴う。

生成 AI の支援により、手動テストケースと自動テストスクリプトの両方を作成し、優先順位を付け、テスト実行スケジュール内に配置することができる。生成 AI は、テストケース、テストデータ、テストスクリプト、テスト環境を含むさまざまなテストウェアの作成と評価を支援することで、この大規模なテスト活動グループを大幅に支援できる。

生成 AI で支援できる典型的なテスト設計およびテスト実装タスクをいくつか示す。

- **テストケースの生成:** 自然言語処理により、生成 AI は機能要件と非機能要件に基づいてドラフトテストケースを作成できる。適切な情報をプロンプトとして与えると、LLM はテストの前提条件と入力、期待結果、カバレッジ基準を提案し、基本的な機能検証から複雑なエンドツーエンドのテストまで、さまざまなテスト目的を満たすテストケースを作成できる

- **テストデータの合成:** 生成 AI は、極端な状況やさまざまなテスト条件をカバーする、本番データに似た、データプライバシーを保護する代表的な合成テストデータを作成できる。この合成テストデータは、機能テストと非機能テストに使用できる。AI によって生成されたテストデータはアプリケーション要件に合わせて調整でき、機密情報を漏らすことなく現実的なシナリオをシミュレートできる
- **自動テストスクリプト生成:** 生成 AI は、構造化テストケースから手動テスト手順と自動テストスクリプトを生成し、テストステップを解釈して、さまざまなテスト自動化フレームワークと互換性のあるコードに変換できる。これらのテストスクリプトは、新しい要件に基づいて更新または拡張できる
- **テスト実行のスケジュールと優先順位付け:** 生成 AI は、テストケースとその相互依存関係を分析し、優先順位、関連するリスク、リソースの可用性、テストの目的に基づいてテスト実行スケジュールを最適化できる

ハンズオン目標 2.2.2a (H2) : プロンプトチェーン、構造化プロンプト、メタプロンプトを使用して AI を用いてユーザーストーリーから機能テストケースを生成する練習をする

この演習では、プロンプトチェーン、構造化プロンプト、およびメタプロンプト技法を使用して、生成 AI を用いてユーザーストーリーから機能テストケースを開発して、徹底したカバレッジを確保することに焦点を当てる。最初のステップは、特定の出力形式に従って、指定された受け入れ基準に基づいて機能テストケースを生成するように AI に指示するプロンプトを作成することである。2 番目のステップは、生成されたテストケースの完全性を検証することである。ここで、プロンプトは、AI にカバレッジを要約した表を生成させることで、各受け入れ基準がカバーされているかどうかを検証する。最後に、3 番目のステップは、エンドツーエンドのテスト手順の作成を支援するメタプロンプトを作成することである。このメタプロンプトは、包括的なエンドツーエンドテストを生成するようにプロンプトを改良し、効果を最大化するための反復的な改善を促進するのに役立つ。

この演習は、テストケースの生成、カバレッジの検証、およびエンドツーエンドのテストでの LLM の使用についての理解を深めるものである。

ハンズオン目標 2.2.2b (H2) : Few-Shot プロンプト技法を使用して、指定されたユーザーストーリーから Gherkin スタイルのテストケースを生成する

この演習では、Few-Shot プロンプトを使用して、指定されたユーザーストーリーから Gherkin スタイルのテストケースを生成する。事前定義された例と Gherkin 構文の確認から始まり、ステップ 1 では、プロンプトに含める n 個の例を選択する。それぞれの例には、ユーザーストーリー、テスト条件、および期待される出力をモデル化するための、予想される Given-When-Then スタイルのテストケースが含まれている。このプロンプトは新しいユーザーストーリーに適用され、元のテスト条件を反映する Gherkin シナリオが生成される。結果が不正確な場合は、プロンプトまたは例を改良する必要がある。

この演習は、Few-Shot プロンプト技法を現実的なテスト設計およびテスト実装タスクに適用する経験を積むのに役立つ。

ハンズオン目標 2.2.2c (H2) : プロンプトチェーンを使用して、特定の優先順位と依存関係を考慮して、特定のテストスイート内のテストケースに優先順位を付ける

この演習では、生成 AI を使用して、関連するリスク分析とテストケース間の依存関係とともに、特定のテストスイート内のテストケースの優先順位付けを改善することに焦点を当てる。このセッションは、リスクベース、カバレッジベース、要件ベースなどのさまざまなテストアプローチの簡単な概要と、特定のテストスイートのレビューから始まる。次に、参加者は、さまざまなテストの優先順位付け戦略に対する実行可能な優先順位付け計画を生成するためのプロンプトの作成に取り組む。プロンプトと指定された入力データに基づく LLM の結果を手動で検証して、LLM の推論のエラーを検出する必要がある。

この演習の目的は、複数基準の推論機能（ここでは、テストケースの優先順位付けで考慮されるさまざまなリスクと依存関係）を必要とするテストタスクで生成 AI を実験することである。

2.2.3 生成 AI による自動リグレーションテスト

新しいイテレーションやリリースが完了するたびに、実行されるリグレーションテストケースの数が増加することが多く、特に継続的インテグレーション/継続的デリバリー (CI/CD) パイプラインでは、テストの実行頻度が高いため、自動化の理想的な候補となる。生成 AI は、自動リグレーションテストスイートの作成、メンテナンス、最適化を支援することで、このプロセスを合理化できる。コードベースの変更に動的に適応し、影響分析を実行することで、生成 AI は、ソフトウェアのどの領域が最近の変更によって最も影響を受ける可能性が高いかを特定し、最も必要な箇所にリグレーションテストの取り組みを集中させることができる。

生成 AI プロンプトで支援できる典型的な自動リグレーションテストとテストレポートアクティビティをいくつか示す。

- **キーワード駆動による自動テストスクリプトの実装:** LLM を使用すると、キーワード駆動のテスト自動化フレームワークに基づいてテストスクリプトを実装できる。事前定義されたキーワードは共通のテストステップを表す。生成 AI は、これらのキーワードを特定のテストケースにマッピングし、テストスクリプトを生成し、テスト担当者やテスト自動化エンジニアの作業を支援する
- **影響分析とテストの最適化:** 生成 AI を使用してコード変更を分析し、高リスク領域を特定できるため、最も必要な箇所を対象としたリグレーションテストが可能になる
- **自己修復テストと適応テスト:** 生成 AI を使用すると、UI または API の軽微な変更を処理するようにテストスクリプトを自動的に調整し、小さな変更による不必要な故障を防ぎ、テストスイートが長期にわたって安定した状態を維持できるようになる
- **自動化されたテストレポートと洞察:** 生成 AI を使用すると、成功メトリクス、失敗、および重要な洞察を含む詳細なテストレポートをタイムリーに生成できる。また、テストの傾向を可視化し、潜在的な故障箇所に関する予測情報を提供するダッシュボードを関係者に提供できる。
- **強化された欠陥レポートと根本原因分析:** 生成 AI は、テストログ、スクリーンショット、およびテスト環境データを含む包括的な欠陥レポートの自動編集を支援できる

これらのアクティビティは、機能および非機能リグレッションテストを含むさまざまなリグレッションテストに適用できる。ただし、テスト担当者は生成 AI が間違いを犯す可能性があることを認識しておく必要がある。したがって、生成された出力は、関連するリスクに応じて慎重にチェックする必要がある（第 3 章を参照）。

さらに、生成 AI は、エンドツーエンドの GUI および API ベースの自動リグレッションテストを支援でき、それぞれに特有の課題とソリューションがある。GUI テストは、ユーザーインターフェースに繰り返し変更が加えられるため、不安定になることがよくある。生成 AI は、動的ロケータや修正されたインタラクションなどの変更を処理するためにテストスクリプトを自動的に適応させ、手動介入の必要性を減らす。API リグレッションテストは、要求/応答形式、エンドポイント、認証の変更などの課題に直面している。生成 AI は、進化する API 仕様にテストスクリプトを自動的に適応させ、多様なテストデータを生成することで、包括的なカバレッジを維持し、手動更新の必要性を減らす。

ハンズオン目標 2.2.3a (H2) : キーワード駆動テストスクリプトを作成および管理するための Few-Shot プロンプトを練習する

この演習では、GUI テスト自動化フレームワークを使用して、特定の Web アプリケーションのテストスクリプトを開発および自動化することに焦点を当てる。この演習は、テストの自動化とテストスクリプトのデバッグという 2 つの主要なセクションで構成されている。演習の最初の部分では、キーワードライブラリのドキュメントの作成、初期テストスクリプトの生成、AI によるこれらのテストスクリプトの検証、追加のテストスクリプトによるカバレッジの拡大に関するガイダンスを提供する。2 番目のパートでは、デバッグ支援に重点を置き、システムプロンプトを使用して、テストスクリプトの確認および修正を行う AI アシスタントを作成する。

この演習では、従来のテスト自動化と AI 支援検証を組み合わせ、キーワード駆動テストスクリプトの作成、保守、デバッグに Few-Shot プロンプトを効果的に使用できる方法を示す。

ハンズオン目標 2.2.3b (H2) : リグレッションテストにおけるテストレポート分析のための構造化プロンプトの作成を練習する

この演習では、構造化されたプロンプトを利用して、リグレッションテストレポートを分析する体系的なアプローチを説明する。このプロセスは、提供されたテスト結果の分析とテスト仕様との比較から始まる。次に、同様の欠陥のクラスタリング、既知の異常リストの維持、および発見事項のクロスチェックに進む。各ステップは、単一の LLM 会話内の次のステップにリンクされている。

段階的なアプローチでは、構造化プロンプトを利用して、リグレッションテストの結果とテストログを実用的な洞察に変換し、リグレッションテストのコンテキストで効果的なテストレポート分析を支援する方法を示す。

2.2.4 生成 AI によるテストモニタリングとテストコントロール

テストモニタリングタスクでは、大量の（非構造化の場合もある）データを取得する必要がある。これらのデータは、多くの場合、生成 AI が分析と合成に役立つテストコントロールツールですでに利用可能である。

生成 AI は、次のような多くのテストモニタリングおよびテスト制御タスクを容易にする。

- **テストモニタリングとメトリクス分析:** 生成 AI は、テストモニタリングの自動化だけでなく、潜在的なリスクを予測し、計画からの逸脱をチームに警告する傾向の分析も容易にする。これにより、チームは常に情報を入手し、品質基準を維持するための措置を講じることができる
- **テストコントロール:** 生成 AI は、必要に応じてテストの優先順位付け、テストスケジュールの調整、リソースの再割り当てに関する洞察を提供することで、テストコントロールを支援する。これにより、テストの柔軟性が維持され、優先度の高い領域に重点が置かれることが保証される
- **テスト完了の洞察と継続的な学習:** 生成 AI は、成功や得られた教訓を明確に示したテスト完了レポートを生成することで、テスト活動を支援できる。これにより、チームはテスト戦略を洗練し、将来のテストプロセスを改善することができる
- **テストメトリクスの視覚化とレポートの強化:** 生成 AI は、動的なダッシュボードと自然言語による概要の作成を支援し、すべての関係者が関連メトリクスに確実にアクセスできるようにする。この支援により、迅速な意思決定に必要な情報が提供され、テストの進行状況が明確に表示される

ハンズオン目標 2.2.4 (H0) : AI がテストデータを基に作成したテストモニタリングメトリクスを確認する

このデモンストレーションでは、生成 AI がテストデータを実用的なテストモニタリングのメトリクスに変換することでテストチームを支援し、情報に基づいた意思決定を容易にする方法を示す。LLM は、テストツールから抽出されたテストデータを処理して、テストの進行状況、欠陥の傾向、カバレッジなどの主要なメトリクスを生成し、潜在的なリスクを明示する。これらの AI によって生成されたメトリクスはダッシュボードに表示され、すべての関係者が理解しやすいように自然言語で要約される。

このデモンストレーションでは、生成 AI がどのようにテストデータを実用的な洞察に変換し、テストチームがテストの進行状況を監視し、品質を管理し、変更に迅速に適應できるようにするかを示す。

2.2.5 ソフトウェアテスト用のプロンプト技法の選択

次の表は、テストタスクの特性に応じた、2.1.2 項 で説明した 3 つのプロンプト技法の適合性を示している。

プロンプト技法	推奨される使用例	主な機能と用途
プロンプトチェーン	各ステップで人による検証を行って精度が求められる複雑なタスク	タスクを小さなステップに分割するため、各テストステップの正確さをチェックする、テスト分析、テスト設計、およびテスト自動化に役立つ
Few-Shotプロンプト	反復的なタスク、または特定/制約された出力形式タスク	例えば、Gherkin スタイルのテストケース（シナリオベースなど）、キーワード駆動テスト、または特定の出力形式でのテストレポートなど、特定のパターンで繰り返し生成するための例を生成 AI に提供する

メタプロンプト	柔軟で動的なタスク。新しいタスクのプロンプトを作成するのに役立つ	LLM がプロンプトを作成する際のガイドとなる、目的と実行するタスクの一般的な説明。テストレポート分析や異常検出など、あらゆる種類の複雑なタスクに役立つ
---------	----------------------------------	--

1 つのユースケースに対して複数の技法を使用することも可能である。例えば、メタプロンプトを使用して初期プロンプトを作成できる。この生成されたプロンプトには、調整が必要で拡張可能な例が含まれている場合がある（**Few-Shot** プロンプト）。最後に、中間ステップを検証できるよう、タスクを小さなサブタスクに分割する（プロンプトチェーン）と便利である。

ハンズオン目標 2.2.5 (H1) : 特定のテストタスクに対してコンテキストに適したプロンプト技法を選択する

この演習では、さまざまなテストタスクに適切なプロンプト技法を選択することに焦点を当てる。参加者には、さまざまな課題を持ついくつかのテストタスクが与えられる。各テストタスクについて、参加者はタスクの性質（精度が必要か、それとも反復構造が必要か）を評価し、コンテキストに最も適合し、タスクの特定のニーズを満たすプロンプト技法を提案する必要がある。選択についてはグループで話し合う。

この演習は、実際のテスト作業でさまざまなプロンプト技法を効果的に使用方法について理解を深めることを目的としている。

2.3 生成 AI の結果を評価し、ソフトウェアテストタスクのプロンプトを改良する

ソフトウェアテストで生成 AI のパフォーマンスを評価するには、生成された出力の品質、関連性、有効性を評価するための明確なメトリクスセットが必要である（Li 2024）。これらのメトリクスは、一般的なものであるかタスク固有のものであるかにかかわらず、LLM プロンプトの最適化に役立つ。

2.3.1 テストタスクにおける生成 AI の結果を評価するためのメトリクス

テストタスクの生成 AI 結果の品質と効率を評価するには、いくつかのメトリクスを使用できる。

メトリクス	説明	例
正確さ	専門家が作成したテストケース、要件、またはその他の基準に対して、生成された出力の全体的な正確さを測定する	生成されたテストケースが指定されたすべての要件をどの程度カバーしているか
精度	特定の目的に関して、生成された出力の正確さを評価する	生成されたテストケースが不正を正しく特定する程度

再現率	データセット内のすべての関連インスタンスを識別するモデルの能力を測定する	生成されたテストケースがデータクラスの有効および無効な同値パーティションをどの程度カバーするか
関連性とコンテキストへの適合性	生成された出力が適用可能であり、特定のコンテキストに適切であるかどうかを判断する	生成されたテストケースがテストベースとどの程度一致し、ドメイン固有の要件を統合しているか
多様性	繰り返しを避けて、幅広い入力とシナリオを確実にカバーする	生成されたテストケースがさまざまなユーザーの行動をどの程度カバーし、エッジケースを調査するか
実行成功率	生成されたテストケースまたはテストスクリプトのうち、正常に実行できる割合を測定する	正常に動作しているテスト環境で、構文エラーや出力形式の問題を発生させることなく、生成されたテストスクリプトを実行できる数を決定する
時間効率	手動テスト作業と比較して節約された時間を評価する	AI がテストケースを生成するのに必要な時間と、人間が同等のテストを手動で作成するのにかかる時間

これらの一般的なメトリクスに加えて、タスク固有のメトリクスを調整して、生成 AI が特定のテストアクティビティをどの程度支援しているかを評価することができる。

これらのメトリクスを効果的に評価するために、テスト担当者は手動レビューを実行したり、LLM の出力を事前定義された参照データと比較するなどの方法で自動化したりできる。生成 AI の非決定的な性質を考慮すると、メトリクスは統計的に関連したデータに基づく必要がある。

ハンズオン目標 2.3.1 (H0) : テストタスクでの生成 AI の結果を評価するためにメトリクスをどのように使用できるかを観察する

特定のテストタスクのデモンストレーション中に、生成 AI の結果を評価するためのタスクに適合したメトリクスと、そのテストタスクで LLM で得られた結果への具体的な適用が示される。

このデモンストレーションは、ソフトウェアテストの生成 AI の結果に信頼性を与えるための評価メトリクスの重要性を示している。

2.3.2 プロンプトを評価し、反復的に改良するための技法

上記のメトリクスに基づいて、プロンプトの評価と改良のための特定の技法を使用して、AI の結果を改善する。

- **反復的なプロンプト修正:** 基本プロンプトから開始し、観察された結果に基づいてそれを反復的に修正し、徐々にコンテキストを追加したり、文言調整（用語などについて）をすることで具体性と関連性を向上させる

- **プロンプトの A/B テスト:** 複数のバージョンのプロンプトを作成し、事前定義されたメトリクスに基づいてどのバージョンがより良い結果を生み出すかを評価する。このアプローチは、どのプロンプトの言い回しまたはプロンプト構造が最も正確で関連性の高い結果を生み出すかを判断するのに役立つ
- **出力分析:** AI によって生成された出力の不正確さや矛盾を検査する（例えばテストベースに関して）。エラーと矛盾の種類を理解すると、プロンプトを改良して、今後の反復で同様の欠陥が発生するのを避けることができる
- **ユーザーのフィードバックを統合する:** 生成された出力の有用性と明確さについてテスト担当者からの意見を収集する（例えば生成されたテストの詳細レベルに関して）。彼らの洞察を分析し、それを使用してプロンプトを改良し、現実世界のテストのニーズをより適切に満たすことができる
- **プロンプトの長さや具体性を調整する:** さまざまなプロンプトの長さや詳細レベルを試すこと。場合によっては、コンテキストを追加すると応答の品質が向上することがある。他の場合には、短いプロンプトの方がより適切な汎化性能が得られる可能性がある

これらの技法を使用することで、テストチームはプロンプトの評価および最適化セッションを組織して、生成 AI プロンプトの継続的な改善を保証できる。テストチームやテスト組織全体で実践を共有することは、プロンプト技法を標準化し、一貫した品質を維持するのに役立つだけでなく、学習と反復的な改善の文化を促進する。この協調的なアプローチは、テストチームが共有された洞察を活用し、同じ誤りを繰り返すことを防ぎ、生成 AI ツールの活用方法を継続的に改善できるようにすることで、生成 AI テスト方法論の進化に貢献する（例えばプロンプトライブラリを共有することによって）。

ハンズオン目標 2.3.2 (H1) : 特定のテストタスクのプロンプトを評価し、最適化する

この演習では、指定されたテストタスクにプロンプト最適化技法を適用することに焦点を当てる。参加者は最初のプロンプトから始めて、それを繰り返し改良して、AI によって生成された結果を改善する。A/B テストや人間による検証などの技術を使用して、プロンプトの品質を評価し、改善する。目標は、反復的な改良がどのようにより効果的で状況に応じたテストケースの生成につながるかを参加者が体験することである。

演習が終了するまでに、参加者はプロンプト改良を数回繰り返し実行し、AI 出力の品質を向上させるために説明したメトリクスを使用して各反復を評価する。

3 ソフトウェアテストにおける生成 AI のリスク管理 - 160 分

キーワード

セキュリティ、脆弱性、データプライバシー

生成 AI 固有のキーワード

ハルシネーション、temperature、推論エラー、バイアス

第 3 章の学習の目的とハンズオン目標:

3.1 ハルシネーション、推論エラー、バイアス

- | | | |
|-------------|------|--|
| GenAI-3.1.1 | (K1) | 生成 AI システムにおけるハルシネーション、推論エラー、バイアスの定義を想起する。 |
| GenAI-3.1.2 | (K3) | LLM 出力のハルシネーション、推論エラー、バイアスを特定する。 |
| HO-3.1.2a | (H1) | 生成 AI でのテストでハルシネーションを実験する。 |
| HO-3.1.2b | (H1) | 生成 AI を使用したテストで推論エラーを実験する。 |
| GenAI-3.1.3 | (K2) | ソフトウェアテストタスクにおける生成 AI ハルシネーション、推論エラー、バイアスを軽減する技法をまとめる。 |
| GenAI-3.1.4 | (K1) | LLM の非決定的動作に対するリコール軽減技法を想起する。 |

3.2 ソフトウェアテストにおける生成 AI のデータプライバシーとセキュリティリスク

- | | | |
|-------------|------|--|
| GenAI-3.2.1 | (K2) | ソフトウェアテストでの生成 AI の使用に関連する主要なデータプライバシーとセキュリティリスクについて説明する。 |
| GenAI-3.2.2 | (K2) | ソフトウェアテストで生成 AI を使用する場合のデータプライバシーと脆弱性の例を挙げる。 |
| GenAI-3.2.3 | (K2) | データプライバシーを保護し、ソフトウェアテスト用の生成 AI のセキュリティを強化するための緩和戦略を要約する。 |
| HO-3.2.3 | (H0) | ケーススタディをテストするために、特定の生成 AI におけるデータプライバシーとセキュリティリスクを認識する。 |

3.3 ソフトウェアテスト用の生成 AI のエネルギー消費と環境への影響

- | | | |
|-------------|------|--|
| GenAI-3.3.1 | (K2) | ソフトウェアテストにおける生成 AI のエネルギー消費に対するタスクの特性とモデルの使用の影響を説明する。 |
| HO-3.3.1 | (H1) | シミュレーターを使用して、生成 AI を使用して特定のテストタスクのエネルギーと CO ₂ 排出量を計算する。 |

3.4 AI の規制、標準、ベストプラクティスフレームワーク

- | | | |
|-------------|------|---|
| GenAI-3.4.1 | (K1) | ソフトウェアテストにおける生成 AI に関連する AI 規制、標準、ベストプラクティスフレームワークの例を想起すること |
|-------------|------|---|

3.1 ハルシネーション、推論エラー、バイアス

生成 AI システム、特に LLM は、ハルシネーション、推論エラー、バイアスなどの特定の欠陥が発生する傾向がある。これらの欠陥により、テストタスクの生成 AI 出力の品質が低下し、その結果、テスト担当者の期待を満たさないテストウェアが生成される。これらのハルシネーション、推論エラー、バイアスは、テスト担当者が LLM 出力で特定する必要がある、これらのリスクを軽減するための措置を講じる必要がある。

LLM の非決定的な動作（1.1.2 項を参照）により、この種の欠陥を修正することが困難になる。1 つの LLM 出力では修正されているように見えるが、同じ LLM との別の会話で再び現れることがある。

3.1.1 生成 AI におけるハルシネーション、推論エラー、バイアス

ハルシネーション（日本語では「幻覚」）は、事実と異なっているか、特定のタスクに無関係であるように見える出力を LLM が生成するときに発生する。ソフトウェアテストでは、ハルシネーションは、LLM が架空または無関係なテストケースを作成したり、誤ったテストスクリプトや機能しないテストスクリプトを生成したり、存在しない受け入れ基準を検証するテストケースを提案したりするといった形で現れることがある。これはテスト担当者に誤解を与え、テスト出力の有効性を損なう可能性がある。

推論エラーは、LLM が因果関係、条件ロジック、段階的な問題解決プロセスなどの論理構造を誤って解釈し、誤った結論を導くときに発生する。人間とは異なり、LLM は真の論理的推論を欠き、パターンマッチングに依存しているため、数学的推論などのタスクを実行するときに誤った論理を引き起こす可能性がある（Mirzadeh 2024）。テスト計画とテストケースの優先順位付けは、論理的推論を必要とし、LLM が推論エラーを起こす可能性があるテストタスクの例である。

LLM バイアス（Gallegos 2024）は、モデルがトレーニングされたデータに由来する。これらのバイアスにより、特定の種類の情報、アプローチ、または仮定を好む結果が生じる可能性がある。例えば、主に英語のデータでトレーニングされた LLM は、英語以外の視点の反映が不十分になる可能性がある。ソフトウェアテストでは、例えばテストデータを生成したり、テストケースの受け入れ基準を詳細化したりするときに、バイアスが LLM の応答に影響を与える可能性がある。

生成 AI 出力のハルシネーション、推論エラー、バイアスは、トレーニングデータの性質とトランスフォーマーモデルの固有の制限に起因する（第 1 章を参照）。これらの課題を認識して対処することで、テストプロセスにおける生成 AI の成果物の品質が向上する。

3.1.2 LLM 出力のハルシネーション、推論エラー、バイアスを特定する

生成 AI システムをソフトウェアテストに効果的に統合するには、LLM 出力のハルシネーション、推論エラー、バイアスを検出する能力が必要である。問題の種類に応じて、さまざまな検出アプローチを適用できる。以下は、レビュー、またはレビューと自動検証の組み合わせを通じて適用される一般的なアプローチである。

ハルシネーションの検出:

- 相互検証: AI によって生成された出力を既存のドキュメント、要件、既知のシステム動作と比較する。自動化ツールは、確立されたデータソースと出力を相互参照して、不一致を検出するのに役立つ

- ドメインの専門知識に関するコンサルティング: 対象分野の専門家に依頼して、生成されたコンテンツの正確性を検証する。彼らの専門知識は、自動化システムでは見落としがちな微妙な洞察を得るために不可欠である
- 一貫性チェック: 生成された出力が相互に一貫しており、既知の情報と一貫していることを確認する。自動化システムは、パターンを特定し、不一致を検出するのに役立つ

推論エラー検出:

- 論理検証: AI が生成したコンテンツについて、論理の流れ（生成されたテキスト内の整合性、首尾一貫性、構造化された推論など）の一貫性および正確性を、レビューサイクルを通じて評価する。
- 出力テスト: 例えば、生成されたテストケースまたはテストスクリプトをテストオブジェクトに対して実行して、テスト結果を検証する。これは、生成されるテストウェアの種類に応じて、部分的または完全に自動化できる

バイアス検出:

- 合成テストデータなど、生成されたテストウェアがテスト戦略に対してどのように公平かつ正確に表現されているかをレビューする
- LLM によって生成された出力で非機能テストが過小評価されているなど、テストタイプに関連するバイアスを評価する

これらの検出方法の実際の実装は、生成 AI で実行されるテストタスクにおけるハルシネーション、推論エラー、またはバイアスの推定リスクレベルに依存する。

ハンズオン目標 3.1.2a (H1) : ソフトウェアテストタスクに関連する生成 AI ハルシネーションの実験

この演習では、ソフトウェアテストの知識体系に関連した生成 AI ハルシネーションの例を実験することに焦点を当てる。参加者は、少なくとも 2 つの LLM が無関係な要素を捏造する状況（例えば、特定のコンテキストデータに存在しない無関係な基準を追加する）に直面することを試みる。プロンプトのバリエーションをテストして、プロンプトがハルシネーションに与える影響を調べる。

この演習では、ソフトウェアテストにおける生成 AI ハルシネーションの特定についての理解を深める。

ハンズオン目標 3.1.2b (H1) : テスト計画タスクにおける生成 AI 推論エラーの実験

この演習では、生成 AI 推論エラーの例を示すことに重点を置いている。テスト計画の分野で解決すべき問題の例（テスト工数の見積もりやテストケースの優先順位付けなど）（[ISTQB_CTFL_SYL] 第 5 章を参照）。この演習は、ある程度複雑な入力データを考慮して設計されており、問題解決スキルが必要であり、この目的での LLM の制限を強調している。LLM の結果は、達成されるべき正確な結果と比較する。3 つの異なる LLM タイプ（LLM、SLM、および推論モデル）が試行し、プロンプトのバリエーションを使用して結果の改善を試みる。

この演習では、論理的な問題解決スキルを必要とするソフトウェアテストタスクにおいて、生成 AI 推論エラーを特定する方法についての理解を深める。

3.1.3 ソフトウェアテストタスクにおける生成 AI ハルシネーション、推論エラー、バイアスを軽減する技法

ソフトウェアテストにおける生成 AI の望ましくない結果を最小限に抑えるために、ハルシネーション、推論エラー、バイアスを減らすためのいくつかの戦略を採用できる。これらの問題は、プロンプトが適切に設計されていない場合（第 2 章を参照）、または特定のテストタスクに関連するコンテキスト入力データが不足している場合に発生する可能性が高くなる。AI のハルシネーション、推論エラー、バイアスに関連するリスクを軽減するための主な技法には、次のようなものがある。

- 完全なコンテキストを提供する: プロンプトにすべての関連情報（2.1.1 項を参照）が含まれていることを確認し、AI が正確な結果を生成するように導くための包括的なコンテキストを提供する
- プロンプトを管理可能なセグメントに分割する: プロンプトチェーン技法（2.1.2 項を参照）を使用して、複雑なプロンプトを小さなステップに分割し、次の出力に進む前に各出力を系統的に検証する。この段階的なアプローチは、生成プロセスの早い段階で推論エラーを検出するのに役立つ
- 明確で解釈可能なデータ形式を使用する: 生成 AI にとって曖昧または解釈が難しい形式は避けること。構造化された単純な形式は、モデルがタスクの重要な側面に焦点を当てるのに役立つ
- タスクに適切な生成 AI モデルを選択する。当面のタスク用に特別にトレーニングされた LLM を使用する（5.1.3 項を参照）
- モデル間で結果を比較する: 必要に応じて、いくつかの LLM でプロンプトを評価し、出力を比較すると、出力エラーを検出し、最も信頼性の高い結果を選択するのに役立つ

第 4 章では、LLM の結果を改善するための 2 つの補完的な技法、RAG とファインチューニングを紹介する。

3.1.4 LLM の非決定的動作の軽減

LLM の固有の非決定的な動作（Shuyin 2023）により、同じ入力提供された場合でも出力が変動する可能性がある。これは、推論中に使用される確率的サンプリングプロセスから生じる。したがって、LLM を使用する場合に一貫性のある再現可能な結果を達成することは、特に長い出力の場合には困難になる可能性があり、ばらつきのリスクが高まる。

完全な再現性は保証できないが、特定の戦略はばらつきを減らすのに役立つ。

- LLM の Temperature パラメータ設定の調整: 応答生成（推論）中に Temperature を下げると、確率分布が狭くなり、ランダム性が減少し、より一貫した出力が得られる。ただし、これにより、応答の創造性と多様性も制限され、出力がより反復的になったり、過度に決定的になったりすることがある
- ランダムシードの設定: 一部の LLM 実装では、乱数ジェネレーターのシード値を設定でき、同じ疑似乱数（つまり、決定論的な乱数値）シーケンスが使用されるようになり、再現性が向上する

LLM 出力におけるハルシネーションや推論エラーのリスクを軽減するには、構造化された一貫した評価プロセスを保証するために出力検証の一部の側面を自動化するなど、この非決定的な動作に対処する必要がある。

3.2 ソフトウェアテストにおける生成 AI のデータプライバシーとセキュリティリスク

テストにおける生成 AI の利用では、LLM を利用したテストインフラストラクチャにおける機密情報の取扱いと潜在的な脆弱性により、データプライバシーとセキュリティに関連するリスクが生じる。セキュリティ侵害、不正アクセス、機密データの漏洩を防ぐには、堅牢なデータ保護が不可欠である。

3.2.1 生成 AI の使用に関連するデータプライバシーとセキュリティリスク

生成 AI は、機密情報や個人を特定できる情報が含まれる可能性のある大量のデータを処理できる。これにより、次のようなデータプライバシーの懸念が生じる。

- 意図しないデータ漏洩: 生成 AI モデルは、機密情報を誤って公開する出力を生成する可能性がある
- データ利用の制御の欠如: 生成 AI ツールは、明示的なユーザーの同意または制御なしに機密データを保存および処理する可能性がある。これにより、悪用や不正アクセスが発生する可能性がある
- コンプライアンスのリスク: 一般データ保護規則 (GDPR、規則 (EU) 2016/679) などのデータ保護規則に準拠せずに生成 AI ツールを使用すると、法的紛争につながる可能性がある

さらに、生成 AI を使用してテストする場合、次のような特定のセキュリティリスクが発生する。

- LLM を利用したテストインフラストラクチャは、データ侵害や不正アクセスなどのセキュリティ攻撃に対して脆弱になる可能性がある
- 悪意のある攻撃者は、操作的な攻撃 (3.2.2 項を参照) など、LLM の脆弱性を悪用して、動作を変更したり、機密情報を抽出したりする可能性がある
- 攻撃者は悪意のある入力データを意図的に導入して LLM を誤解させ、その精度やセキュリティを損なう可能性がある

3.2.2 テストプロセスおよびツール向け生成 AI におけるデータプライバシーと脆弱性

次の表に、生成 AI テストプロセスおよびツールにおける攻撃ベクトルの例をいくつか示す。

攻撃ベクトル	説明	例
データ漏洩	機密トレーニングデータを抽出することを目的としたリクエストの送信	AI のメモリに過負荷をかけるための長いプロンプトで LLM コンテキストウィンドウを超えると、トレーニングデータのランダムな断片が明らかになり、機密情報が漏洩する可能性がある
リクエストの操作	AI の出力を混乱させるデータの導入	AI を別のコンテキストに誘い込み、受け入れ基準などに関するハルシネーションを引き起こす画像
データポイズニング	トレーニングデータの操作	AI が生成したテストレポートの結果を評価する際に、虚偽の評価を提供する
悪意のあるコードの生成	LLM を操作して、使用中にバックドア（外部コマンド呼び出しなど）を生成する	特定の悪意のある IP との通信チャネルを開くコードの生成

3.2.3 生成 AI を用いたテストにおけるデータプライバシー保護とセキュリティ強化のための緩和策

生成 AI が主流になり、固有のリスクが伴うため、それらを軽減するための規制や基準が生まれている（3.4.1 項を参照）。

GDPR などのデータ保護規制は、生成 AI の適用を明示的に制限しないが、特にデータの収集、処理、保存の目的の合法性と制限に関して、実行できることを制限する可能性のある保護手段を提供する。

これらのリスクを軽減するには、組織は次のような堅牢なデータプライバシー対策を実装する必要がある。

- データの最小化: 法的に許可されない限り機密データの処理を回避し、AI テストで必要な量の非機密データのみを使用して、データプライバシーのリスクを軽減する
- データの匿名化と仮名化: 機密情報をマスクしたり、識別不可能なデータで置き換えたりする
- 安全なデータの保存と送信: 強力な暗号化とアクセス制御を実装する
- リソースのトレーニング: 組織は、生成 AI ツールの責任ある使用を保証し、倫理的実践を促進し、潜在的なリスクを軽減するために、明確なトレーニングプログラムとポリシーを確立する必要がある

テスト用に生成 AI を実装する場合は、追加の緩和戦略を検討できる。

- 生成された出力の体系的なレビュー: 生成 AI を利用したテストタスクの品質と精度を確保するには、人間による評価が不可欠である

- 別の LLM との比較による評価: これには、特定のタスクで複数の LLM を使用し、それらの応答を比較することによって出力を評価することが含まれる
- 安全な運用環境の選択: 必要な機密性のレベルに応じて、組織はさまざまな安全なソリューションを選択できる。LLM プロバイダーが提供する商用の安全な製品を使用するか、安全なクラウドで LLM を運用するか、組織のインフラストラクチャに LLM をインストールする
- 定期的なセキュリティ監査と脆弱性評価: 生成 AI システムの弱点を特定して対処する
- セキュリティのベストプラクティスを常に最新の状態に保つ: 最新のセキュリティガイドラインと技術を常に把握する

これらの戦略は多くの場合相互に補完的であり、生成 AI の使用中にデータのセキュリティを確保するには、これらの戦略を組み合わせる必要がある。組織内に上級セキュリティエンジニア、法律顧問、最高技術責任者（CTO）、または最高情報セキュリティ責任者（CISO）がいる場合は、その者に関与してもらうことを強く推奨する。

ハンズオン目標 3.2.3 (H0) : ケーススタディをテストするために、特定の生成 AI におけるデータプライバシーとセキュリティリスクを認識する

このデモンストレーションでは、ソフトウェアテストで生成 AI を使用するときデータプライバシーとセキュリティのリスクがどのように発生する可能性があるかを示す。参加者はケーススタディを検討して、モデルの脆弱性、不正なデータアクセス、生成された出力の悪意のある使用などの潜在的な脅威を特定する。彼らは、倫理的および実質的な影響を考察しながら、安全なデータ処理、堅牢なアクセス制御、AI モニタリングの実践などの緩和戦略を検討する。

最後までに、参加者はデータプライバシーの原則を理解し、生成 AI を用いたテスト条件におけるセキュリティリスクを認識して対処する方法を学ぶ。

3.3 ソフトウェアテストにおける生成 AI のエネルギー消費と環境への影響

(Luccioni 2024a) などの研究では、LLM のトレーニングと処理には、多数の特殊なコンピューティングリソースを集中的に使用することが示されている。LLM は Web ベースのサービスとして利用可能であり、その使用によりデバイス、ネットワーク、データセンターの負荷が増加し、エネルギー消費量の増加につながる。

3.3.1 生成 AI の使用がエネルギー消費と CO2 排出量に及ぼす影響

使用量が増加するとエネルギー消費量が急激に増加するため、生成 AI の環境への影響を過小評価してはならない。タスクの複雑さと必要な計算リソースは、エネルギー消費に影響する。例えば、強力な AI モデルを使用して 1 つの画像を生成すると、スマートフォンをフル充電するのと同じくらい多くのエネルギーを消費する可能性があるが、テキストの生成ではスマートフォンの充電量のわずかな割合しか消費しない (Heikkilä 2023)。

生成 AI の環境への影響に関する正確なデータを入手するのは困難でも (Luccioni 2024b)、これらのエネルギー集約型の作業が集散的に多大な CO₂ 排出に寄与していることは明らかである (Berthelot 2024)。単一の検索やテキスト生成のタスクは取るに足らないように思えるかもしれないが、世界中の何百万人ものユーザーが繰り返すことで、その累積的な影響は環境に大きな負担を与える。

生成 AI によってもたらされる環境リスクを軽減するには、不必要なモデルの相互作用を制限するなどのベストプラクティスを採用することが重要である。

ハンズオン目標 3.3.1 (H1) : シミュレーターを使用して、生成 AI を使用して特定のテストタスクのエネルギーと CO₂ 排出量を計算する

この演習では、ソフトウェアテスト内のさまざまな生成 AI タスクのエネルギー消費とそれに伴う CO₂ 排出量の評価に焦点を当てる。参加者はシミュレーションを使用してこれらのメトリクスを計算し、さまざまなタスクの特性とモデルの使用法が環境にどのような影響を与えるかを調査する。

さまざまな要因がエネルギー消費と排出量にどのように影響するかを観察することで、参加者は LLM 利用時のエネルギー消費量を左右する要因を理解する。

3.4 AI の規制、標準、ベストプラクティスフレームワーク

生成 AI は、さまざまなテストタスクでテスト担当者を支援することにより、ソフトウェアテストを変革している (第 2 章を参照)。ただし、これらの機会は、推論エラー、データプライバシー、脆弱性、環境への影響などの重大なリスクももたらす (3.1 節、3.2 節、および 3.3 節を参照)。これらのリスクに対処するには、AI に関する一般的な規制、標準、ベストプラクティスのフレームワークを考慮する必要がある。

3.4.1 ソフトウェアテストにおける 生成 AI に関連する AI 規制、標準、およびフレームワーク

以下は、ソフトウェアテストでの生成 AI の使用に関連する主要なガイドラインの概要である。

名前/タイプ	説明	ソフトウェアテストへの応用
ISO/IEC 42001:2023 情報技術 - 人工知能 - 管理システム タイプ: 標準	組織内で AI システムを管理するための要件を定める	テストにおける生成 AI が推奨プラクティスに準拠していることを確認し、一貫性と信頼性を促進する

ISO/IEC 23053:2022 機械学習を使用した人工知能 (AI) システムのフレームワーク タイプ: 標準	AI ライフサイクルプロセスのフレームワークを提供し、フォールトレランスと透明性を強調する	生成 AI をテストに使用する場合のデータ品質、透明性、フォールトレランスのフレームワークを提供する
EU AI 法 タイプ: 規制	AI リスクに対処する法的枠組みを確立し、アプリケーションをリスクレベルごとに分類する 出典: (AI 法 2024)	テストで使用される生成 AI の透明性、説明責任、バイアス軽減に関するコンプライアンスを義務付ける
NIST AI リスク管理フレームワーク (米国) タイプ: フレームワーク	公平性、透明性、セキュリティに重点を置いて、AI リスクを管理するためのガイドラインを提供する 出典: (NIST AIRMF 1.0)	生成 AI における公平性を確保し、リスクを緩和することで、バイアスのあるテスト結果を防止する

AI テクノロジーとその規制環境は進化し続けるため、テスト組織は、この表にあるような規制、標準、国内法、ベストプラクティスフレームワークの動向を常に把握することが不可欠である。

4 LLM を利用したソフトウェアテスト用テストインフラストラクチャ - 110 分

キーワード

テストインフラストラクチャ

生成 AI 固有のキーワード

ファインチューニング、LLM 搭載エージェント、LLMOps、RAG、ベクトルデータベース

第 4 章の学習の目的とハンズオン目標:

4.1 LLM を利用したテストインフラストラクチャのアーキテクチャアプローチ

- | | | |
|-------------|------|--|
| GenAI-4.1.1 | (K2) | LLM を利用したテストインフラストラクチャの主要なアーキテクチャコンポーネントと概念について説明する。 |
| GenAI-4.1.2 | (K2) | RAG を要約する。 |
| HO-4.1.2 | (H1) | 特定のテストタスクに対して RAG を試してみる。 |
| GenAI-4.1.3 | (K2) | テストプロセスの自動化における LLM 搭載エージェントの役割と応用について説明する。 |
| HO-4.1.3 | (H0) | LLM 搭載エージェントが反復的なテストタスクの自動化をどのように支援するかを観察する。 |

4.2 ファインチューニングと LLMOps: ソフトウェアテスト用の生成 AI の運用化

- | | | |
|-------------|------|---|
| GenAI-4.2.1 | (K2) | 特定のテストタスクに対する言語モデルのファインチューニングについて説明する。 |
| HO-4.2.1 | (H0) | 特定のテストタスクと言語モデルのファインチューニングプロセスの例を観察する。 |
| GenAI-4.2.2 | (K2) | LLMOps と、テストタスク用の LLM の展開と管理におけるその役割について説明する。 |

4.1 LLM を利用したテストインフラストラクチャのアーキテクチャアプローチ

AI チャットボットと LLM を利用したテストツールは、LLM を使用する 2 種類のテストインフラストラクチャである (1.2.2 項を参照)。

LLM を利用したテストインフラストラクチャ (4.1.1 項を参照) の基本アーキテクチャを超えて、RAG (4.1.2 項を参照) および LLM 搭載エージェントアーキテクチャ (4.1.3 項を参照) は、ソフトウェアテストで LLM を使用する機能と有用性を拡張する。

4.1.1 LLM を利用したテストインフラストラクチャの主要なアーキテクチャコンポーネントとコンセプト

LLM を利用したテストインフラストラクチャとは、自動化、推論、意思決定を強化するために LLM をソフトウェアテストプロセスに統合するシステムを指す。主に会話形式のやりとりに焦点を当てた従来の AI チャットボットとは異なり、LLM を利用したテストツールは、テスト関連のクエリの処理、要件の分析、テストケースの生成、出力の評価によってソフトウェアテストを支援するように設計されている。

LLM を利用したテストインフラストラクチャの一般的なアーキテクチャは、LLM との安全かつ効率的な対話を促進するマルチコンポーネント設計に従っている。このアーキテクチャは、フロントエンドコンポーネントとバックエンドコンポーネント、外部データソース、統合 LLM で構成される。

- フロントエンドは、テスト担当者がクエリやコマンドを入力してシステムと対話するユーザーインターフェースとして機能する
- バックエンドはユーザー入力を処理し、認証、データ取得、プロンプト準備、LLM との対話などの重要な機能を管理する
- LLM は、サードパーティサービス (API 経由でアクセス) またはカスタム社内モデルとしてホストされ、構造化されたプロンプトに基づいて応答を生成する

このアーキテクチャは、LLM や複数のデータソースを持つバックエンドなどのインテリジェントな処理コンポーネントを組み込むことで、従来のクライアント/サーバーモデルを超えている。

1. LLM は単なるサーバーではなく、テスト成果物に基づいて解釈および推論を行うスマートな処理コンポーネントである
2. スクリプト化された応答に従うルールベースのチャットボットとは異なり、LLM を利用したテストインフラストラクチャは、要件、コード、テスト結果などのコンテキストからテストの洞察を動的に生成する
3. バックエンドは、次のような複数のデータソースを統合する
 - リレーショナルデータベース (テストケースなど、テストで使用される構造化データ用)
 - ベクトルデータベース (エンベディングを使用した関連コンテンツのセマンティック検索用。4.1.2 項を参照)

4. バックエンドは後処理を通じて LLM の生の出力を強化し、その応答がフロントエンドに提示される前にテストプロセスのテスト条件と一致していることを確認する

4.1.2 RAG

RAG (検索拡張生成) は、応答生成プロセスに追加のデータソースを組み込むことで LLM を強化し (Zhao 2024)、それによって出力の関連性と精度が向上する。

RAG は、検索システムと言語モデルを組み合わせ、コンテキストを認識した応答を生成する。前処理中に、大きなドキュメントはより小さなチャンク (例: 256 ~ 512 トークン) に分割され、集中的な検索とモデルコンテキストウィンドウとの互換性が保証される。各チャンクは、事前トレーニングされたモデルを使用してクリーンアップされ、処理され、高次元ベクトル (エンベディング) にエンコードされる。これらのエンベディングはベクトルデータベースに保存できるため、実行時 (推論) での類似性に基づく効率的な検索が可能になる。ユーザークエリがエンコードされ、意味論的な類似性に基づいて関連するチャンクが検索され、これらのチャンクは言語モデルのコンテキストとして使用され、根拠のある応答が生成される。

関連する応答は本質的に、検索プロセス中に収集された、関連性があり、正確で、文脈上適切な情報に深く根ざした言語モデルによって生成される出力である。これにより、応答がモデルの既存のトレーニングに基づくだけでなく、プロンプトに関連する正確なデータで強化されることが保証される。検索と生成の間のこの相乗効果により、応答の精度と関連性が向上し、応答の信頼性が高まり、ユーザーにとって有益なものになる。

ユーザープロンプト処理フェーズでは、RAG システムは 2 段階のプロセスを通じて動作する。

1. 検索: ユーザーのクエリが与えられると、システムは以前に作成されたベクトルデータベースから関連情報を検索する。この取得は通常、プロンプトのエンベディングとチャンクのエンベディングとの間の意味上の類似性に基づいている
2. 生成: 検索された情報は LLM に供給され、LLM は既存の知識と新しく検索したデータを組み合わせた応答を生成し、その結果、より正確で状況に応じた適切な出力が得られる

ソフトウェアテストにおいて RAG を使用すると、LLM を利用したテストインフラストラクチャがデータベース、ドキュメント、リポジトリなどの企業のエンタープライズデータソースにアクセスしてコンテキスト情報をリアルタイムで取得できるようになり、テスト分析やテスト設計などのテストタスクが最新の仕様、要件、既存のテストデータと確実に一致するようになる。

ハンズオン目標 4.1.2 (H1) : 特定のテストタスクに対する RAG の実験

この実践的な演習では、特定のテストタスクに対する RAG 技法の適用に焦点を当てる。参加者は、ドキュメントを組み込んで RAG システムを実験し、複雑な情報に基づいてどの程度正確な回答が生成されるかを観察する。参加者は、指定されたテストタスクで RAG を使用した場合と使用しない場合で LLM の出力を比較する。この演習は、さまざまな種類のテストタスクを処理する際の RAG システムの長所と限界を特定することを目的としている。

検索したデータと生成された結果を調べることで、参加者は LLM を活用したテストプロセスの強化における RAG の役割について洞察を得ることができる。

4.1.3 テストプロセスの自動化における LLM 搭載エージェントの役割

LLM 搭載エージェント (Wang 2024) は、LLM を利用した特殊な生成 AI アプリケーションであり、定義されたタスクの半自律または自律処理用に設計されている。これらのエージェントの中核は、自然言語の理解と生成のために LLM に依存しており、命令を処理し、コンテキストを取得し、知的に行動する機能で補完されている。

質問と回答のやりとりのみに焦点を当てた従来の AI チャットボットとは異なり、LLM を利用したエージェントは、一般に「ツール」と呼ばれる事前定義された一連の機能を呼び出すことでタスクを実行したり、「行動」したりできる。この機能により、外部システムと対話して操作できるため、タスクの実行において非常に汎用性が高くなる。LLM を利用したエージェントの自律性の程度は次のように異なる。

- 自律エージェントは独立して動作し、事前定義されたルール、強化学習、適応的なフィードバックループを使用して人間の介入を最小限に抑えてタスクを実行する
- 半自律エージェントは、定期的に人間の監視を受けながらタスクを実行し、出力がユーザー定義の目標を確実に満たすようにする

マルチエージェントアーキテクチャでは、それぞれが専門的な役割を担う複数のエージェントで構成される協調型システムを採用しており、各エージェントが互いに通信し、連携することで、単一のエージェントよりも効率的に複雑な問題を解決する。複数の AI エージェントによるこのような連携は、「オーケストレーション」として知られている。

テストプロセスでは、LLM 搭載エージェントは人間の推論と意思決定を模倣することでテストタスクを自動化できる。ただし、これらのエージェントは、LLM を使用するときには観察される、ハルシネーション、推論エラー、バイアスの可能性という同じ問題に悩まされている (3.1 節を参照)。これらのエージェントは、不正確または誤解を招く結果を生成する可能性があり、自動テストプロセスの信頼性が低下する可能性がある。これらのリスクは、エージェントの結果に対する自動検証手順を実装するか、重要なタスクに半自律エージェントを使用することで軽減できる。

ハンズオン目標 4.1.3 (H0) : LLM 搭載エージェントが反復的なテストタスクの自動化をどのように支援するかを観察する

このデモンストレーションは、LLM 搭載エージェントによって実行されるテストタスクに焦点を当てる。エージェントに渡される入力データ、エージェントの動作、アクションの結果を示し、エージェントベースのソリューションをテストプロセスに統合するさまざまな側面を示す。

このデモンストレーションでは、テストタスクのコンテキストにおける LLM 搭載エージェントの具体的な例を示す。

4.2 ファインチューニングと LLM Ops: ソフトウェアテスト用の生成 AI の運用化

LLM を利用したテストインフラストラクチャをテスト用に運用するための 2 つの重要な実践には、LLM のファインチューニングと LLM Ops による運用パイプラインの管理が含まれる (Mallach 2024)。

4.2.1 テストタスクに対する LLM のファインチューニング

ファインチューニングでは、LLM や小規模言語モデル (SLM、1.1.2 項 を参照) などの事前トレーニング済み言語モデル (LM) を適応させて、特定のタスクを実行したり、特定のドメインに合わせて調整したりする (Parthasarathy 2024)。これには、対象のデータセットでモデルをさらにトレーニングし、ドメイン固有の知識とニュアンスを学習できるようにすることが含まれる。ファインチューニングすることにより、モデルのパフォーマンスが特殊なアプリケーション向けに強化され、モデルがより正確になり、意図されたユースケースに関連したものになる。

実際には、ファインチューニングは、汎用 LLM に特定の分野に関連する特殊な推論能力を装備したり、その分野に固有の語彙を採用したりするのに適している。ファインチューニングは、リソースの消費量が少ない、SLM として知られる小型のモデルにも適用できる。SLM をファインチューニングすることで、LLM と同じ計算オーバーヘッドを発生させることなく、特定のタスクでより高いパフォーマンスレベルを達成できる。この比較は、タスクの特定の要件に基づいて LLM と SLM の両方を使い分けることの柔軟性と効率性を強調している。

例えば、ソフトウェアテストでは、ファインチューニングにより、LLM または SLM が、組織のコンテキストに固有の出力形式でユーザーストーリーからテストケースを生成できるようになる。組織のユーザーストーリーと対応するテストケースに基づいてモデルをトレーニングすることにより、モデルは組織固有のテストプロセスと用語に適合するようになる。

ソフトウェアテスト用に生成 AI モデルをファインチューニングするには、いくつかの課題がある。

- 高品質でタスク固有のトレーニングデータセットを使用することで、偏った結果や不正確な結果を回避する
- さまざまなシナリオにわたって汎化性能を維持するために、過学習（モデルがトレーニングデータに特化しすぎて、新しい未確認データのパフォーマンスに悪影響を与える）を軽減する
- デバッグや検証を困難にする、モデルの推論における不透明性（LLM がどのように意思決定を行い、出力を生成するかが明確でないこと）に対処する
- ファインチューニングプロセス（LLM の場合）に必要な大量の計算リソースを管理する

ハンズオン目標 4.2.1 (H0) : 特定のテストタスクと LLM/SLM のファインチューニングプロセスの例を観察する

このデモンストレーションでは、特定のテストタスクに合わせて LLM をファインチューニングする際のさまざまな手順を示す。まず、適切な LLM または SLM を選択する。次に、特定のテストタスクに合わせて調整されたデータセットが提示される。次に、ファインチューニングプロセスの例示的なソリューションが示される（例: 機械学習フレームワーク）。最後に、調整済みのモデルにプロンプトを送信し、生成された出力の品質について議論する。

テストタスクの LLM/SLM ファインチューニングプロセスのこのデモンストレーションは、このプロセスのいくつかの重要な側面を示し、特にトレーニングデータの品質に焦点を当てる。

4.2.2 ソフトウェアテスト用に LLM をデプロイおよび管理する場合の LLMOps

LLMOps (Large Language Model Operations) とは、運用環境での LLM の開発、デプロイ、保守を合理化するために設計された一連のプラクティス、ツール、およびプロセスを指す (Sinha 2024)。

組織のテストプロセスでの生成 AI の使用は、いくつかの異なる方法で実現でき、LLMOps の決定に影響を与える。考えられるアプローチは次の 3 つである。

- **AI チャットボットの使用:** このアプローチの主な考慮事項には、コストを最適化しながらデータプライバシーとセキュリティリスクを管理することが含まれる。組織は、必要な保証が与えられている場合は **LLM-as-a-Service** プラットフォームを使用するか、オープンソースライセンスの LLM を使用して社内インフラストラクチャを導入してより高度な制御を行うことができる。データのプライバシーとセキュリティのリスク (3.2 節 を参照) を軽減し、運用効率を確保するには、ベンダーの保証内容または内部の能力を厳密に評価することが重要である
- **生成 AI 機能を備えたテストツールの使用:** このアプローチには、データプライバシー、セキュリティ、運用コストなど、AI チャットボットと同様の考慮事項がある。さらに、組織はテストツールプロバイダーが提供するデータセキュリティとパフォーマンスの保証を評価する必要がある。これらのテストツールは通常、既存のテストプロセスを補完するものであり、十分な費用対効果分析とリスク評価が必要である。
- **生成 AI を活用したテストツールの社内開発:** このアプローチでは、データプライバシーとセキュリティリスクの包括的な制御に重点を置き、計算リソース、データストレージ、スタッフのトレーニングなどの AI 運用コストを慎重に計画する。組織は、生成 AI に固有の開発を検証および維持するための構造化されたプロセスも確立する必要がある。社内ソリューションの開発には、LLM を利用したテストインフラストラクチャの実装とデプロイに関する専門知識が必要である

組織は一部のタスクに AI チャットボットを利用し、他のタスクにはカスタムツールを開発する可能性があるため、これらのアプローチは相互に排他的ではない。したがって、関係する特定のテスト活動に応じて、それらを同時に実装することができる。さらに、RAG や LLM/SLM のファインチューニングなどの追加テクノロジーを組み込むことで、生成 AI によるテストプロセスの有効性と適応性を高めることができる。

5 テスト組織への生成 AI の導入と統合 - 80 分

キーワード

なし

生成 AI 固有のキーワード

シャドーAI

第 5 章の学習の目的とハンズオン目標:

5.1 ソフトウェアテストにおける生成 AI の導入のロードマップ

- | | | |
|-------------|------|---|
| GenAI-5.1.1 | (K1) | シャドーAI のリスクを想起する。 |
| GenAI-5.1.2 | (K2) | ソフトウェアテストの生成 AI 戦略を定義する際に考慮すべき重要な側面について説明する。 |
| GenAI-5.1.3 | (K2) | 特定のコンテキストでソフトウェアテストタスクに LLM/SLM を選択するための主要な基準を要約する。 |
| HO-5.1.3 | (H1) | 特定のテストタスクで生成 AI を使用する場合の経常コストを見積もる。 |
| GenAI-5.1.4 | (K1) | テスト組織での生成 AI の導入における主要な段階を想起する。 |

5.2 ソフトウェアテストに生成 AI を採用する場合の変更の管理

- | | | |
|-------------|------|--|
| GenAI-5.2.1 | (K2) | テストプロセスで生成 AI を効果的に使用するためにテスト担当者に必要な必須のスキルと知識領域を説明する。 |
| GenAI-5.2.2 | (K1) | テスト活動における生成 AI の導入を支援するために、テストチーム内で AI スキルを育成するための戦略を想起する。 |
| GenAI-5.2.3 | (K1) | 生成 AI を導入する際に、テスト組織内でテストのプロセスと責任がどのように変化するかを認識する。 |

5.1 ソフトウェアテストにおける生成 AI の導入のロードマップ

生成 AI を使用したテスト戦略では、達成すべきテスト目標、適切な LLM の選択、プロンプトに使用される入力データの問題、AI の標準および規制への準拠などの重要な側面を慎重に考慮する必要がある。この戦略に基づいて、組織はロードマップを確立し、生成 AI をテストプロセスに統合する進行状況を監視できる。

5.1.1 シャドーAI のリスク

シャドーAI は、セキュリティ、コンプライアンス、データプライバシーに関するリスクを引き起こす可能性がある。

- 情報セキュリティとデータプライバシーの弱点: 個人用 AI ツールには堅牢なセキュリティが欠けている可能性があり、潜在的なデータ侵害につながる可能性がある
- コンプライアンスと規制の問題: 未承認の AI ツールを使用すると、業界標準と規制 (3.4.1 項を参照) への違反につながり、法的責任が生じる可能性がある
- 曖昧な知的財産: 不明確なライセンス契約で AI ツールを使用すると、特に著作権で保護されたデータが適切な許可なしに処理された場合、LLM ユーザーが知的財産紛争にさらされる可能性がある

生成 AI を統合および展開するための戦略と手順は、テスト組織がシャドーAI のリスクを回避するのに役立つ。

5.1.2 ソフトウェアテストにおける生成 AI 戦略の重要な側面

テストで生成 AI 戦略を成功裏に実装するには、組織はいくつかの重要な要素を慎重に考慮して、スムーズな統合と最適な結果を確保する必要がある。これは、テストの生産性の向上、テストサイクルの短縮、テスト品質の向上など、生成 AI の測定可能なテスト目標を定義することから始まる。適切な LLM を選択することは重要であり (5.1.3 項を参照)、既存のテストインフラストラクチャとの互換性を確保し、システムのスケーラビリティ要件を満たしながら、これらのテストの目的に合わせて選択する必要がある。

LLM を利用したテストの有効性は、堅牢なセキュリティ手順で保護された正確で関連性の高い入力データに依存するため、データ品質は重要な役割を果たす。したがって、正確であると信頼できる結果を得るには、高品質の入力データを維持することが重要である。

生成 AI ツールを効果的に使用するために必要な技術的および倫理的スキルをテストチームが確実に身につけるために、包括的なトレーニングプログラムを提供する必要がある。トレーニングに加えて、生成 AI 結果の有効性を測定するために特定のメトリクスを収集する必要がある (2.3.1 項を参照)。

規制基準への準拠と倫理ガイドラインの遵守を確実にするために、組織は、機密データの使用規則、透明性義務 (生成 AI を使用して生成された内容など)、生成されたテストウェアのレビューによる品質ゲートなど、生成 AI の使用に関するプロセスガイドラインを確立する必要がある。

5.1.3 ソフトウェアテストタスク用の LLM/SLM の選択

LLM/SLM には幅広い種類があり、それぞれに異なる機能能力 (例: マルチモーダル入力、推論機能)、技術的特徴 (例: コンテキストウィンドウサイズ)、およびライセンスの種類 (例: 商用ソースとオープン

ンソース) がある。自然言語処理、コード生成、画像分析などのタスクの LLM/SLM を評価するベンチマークは数多くあるが、特にソフトウェアテストタスクに焦点を当てたベンチマークはわずかである (Wenhan 2024)。

したがって、テストタスクに LLM/SLM を選択するには、いくつかの重要な基準を慎重に考慮する必要がある。

- モデルのパフォーマンス: 2.3.1 項 で示されているようなメトリクスを使用して、組織のベンチマークに対して対象のテストタスクに対するモデルのパフォーマンスを評価する
- ファインチューニングの可能性: ドメイン固有のデータを使用して言語モデル (LLM または SLM) をファインチューニングして、特定のユースケースのパフォーマンスを向上させ、特殊なコンテキストでの精度と関連性を高めることが可能かどうか、また有用かどうかを評価する
- 経常コスト: LLM/SLM の使用にかかる経常コスト (ライセンス料や運用経費など) を考慮し、対象のテストタスクに対する組織の予算内に収まるようにする
- コミュニティと支援: 実装やトラブルシューティングを円滑に進めるため、活発なコミュニティ支援と詳細なドキュメントが整っているモデルを選択する

これらの基準を慎重に評価することで、テスト組織は、自組織固有のニーズや制約に適した 1 つ以上の LLM/SLM を選択できる。

ハンズオン目標 5.1.3 (H1) : 特定のテストタスクで生成 AI を使用する場合の経常コストを見積もる

この演習では、さまざまな仮定に基づいて、特定のテストタスクに生成 AI を使用する場合の経常コストを見積もることに焦点を当てる。これには、入力データと出力データ内のトークンの数、使用されるプロンプト、タスクの頻度などの要素が含まれる。少なくとも 1 つの商用ソリューションと 1 つのオープンソースライセンスモデルを含む、複数の LLM/SLM ベンダーの価格モデルを調査および比較する。

この演習では、実用的なテスト条件を使用して生成 AI の経常コストを計算および実験する機会が提供され、さまざまなアプローチやベンダーの財務上の影響を理解するのに役立つ。

5.1.4 ソフトウェアテストへの生成 AI 導入におけるフェーズ

テスト組織内で生成 AI を導入するには、次の 3 つの重要なフェーズが必要である。

1. 発見: 最初のフェーズでは、理解の促進と能力の育成に重点を置く。アクティビティには、生成 AI の概念に関するテストチームのトレーニング、LLM/SLM へのアクセスの提供、テスト担当者を生成 AI に慣れさせて自信を高めるための初期ユースケースの実験が含まれる
2. 開始と使用法の定義: 基本的な理解を確立したら、第 2 フェーズでは、ソフトウェアテストにおける生成 AI の実際の使用例の特定と優先順位付けに焦点を当てる。このフェーズには、LLM を利用したテストインフラストラクチャの評価、専門知識の開発、組織のニーズとの整合性の確保を含む ([ISTQB_CTFL_SYL] 6 章を参照)

3. 利用と反復: この高度な段階では、組織は生成 AI をテストプロセスに完全に統合する。ソフトウェアテストにおける生成 AI および関連ツールの進展を継続的に監視するとともに、持続的な効果と拡張性を確保するため、変革の進捗を測定・管理する仕組みが整備されている。

これらのフェーズは、さまざまなユースケースに合わせて並行して実行できる。例えば、テストの自動化が初期段階にある一方で、テストレポート分析はロードマップ上でより進んだ段階にある場合がある。また、採用やチームの士気に影響を与える可能性のある、離職の恐怖などの懸念を早期に認識して対処することも重要である。

5.2 ソフトウェアテストに生成 AI を導入する際の変更を管理する

テスト組織で生成 AI を適切に導入するには、変更管理プロセスに対する構造化されたアプローチが必要である。重要な側面には、必須の生成 AI スキルの構築と、AI 対応のテストプロセスを採用するために従来のテストの役割を進化させることが含まれる。この変革には、技術的スキルと組織的側面の両方が関係する。

5.2.1 生成 AI を使用したテストに必要なスキルと知識

生成 AI をテストにうまく統合するには、プロンプトエンジニアリング技法を習得し、モデルコンテキストウィンドウを理解し、テストレビュー方法を開発する必要がある。テスト担当者は、テストケースの生成、欠陥レポートの分析、テストデータの生成などのタスクで LLM 主導のテストを評価するには、ドメインおよびテストの専門知識と AI スキルを組み合わせる必要がある。

主要な能力には、LLM の機能の評価、プロンプト改良技術の理解、AI 生成のテストウェアの評価が含まれる。必須の知識には、生成 AI に固有のリスクの理解と、一般的な軽減策の認識が含まれる。テスト担当者は、LLM とテストウェアを共有することによるデータセキュリティへの影響を理解し、適切なデータサニタイズ（機密情報、個人情報、または機密情報の削除またはマスキング）を実装し、データプライバシーを保護するプロンプトエンジニアリングプラクティスに従う必要がある。環境への配慮には、モデルの選択と使用パターンを最適化して計算オーバーヘッドを削減すること、テストタスクに適切なサイズのモデルを選択すること、生成 AI 自動化の利点とコストとエネルギー消費への影響のバランスを取ることが含まれる。

5.2.2 テストチームで生成 AI 機能を構築する

生成 AI を用いたテストのためにテストチームを戦略的にトレーニングするには、実践的なアプローチが不可欠である。これには、さまざまな LLM/SLM で練習し、構造化された学習パスに従い、組織内で共有することで徐々にノウハウを開発することが含まれる。トレーニングの焦点は、ガイド付き演習、ピアラーニング、毎日のテストタスクへの AI の段階的な統合を通じて実践的なスキルを開発することにある。

テストチームのメンバーは、基本的なプロンプト作成の習得から、テスト固有のプロンプトなど、より焦点を絞った技法の使用に進む。プロンプトパターンは、生成 AI を一貫性のある信頼性の高い出力に導くための効果的なプロンプトを作成するための再利用可能なテンプレートである。社内の実践コミュニティは、成功した生成 AI アプリケーションを紹介し、課題を議論し、ベストプラクティスを洗練するための定期的な会議により、継続的な知識の共有を支援する。これらのコミュニティは、プロンプトパターンライブラリを共有し、プロジェクトやドメイン全体でテスト実装のために生成 AI から学んだ教訓を文書化することで、継続的な改善を促進する。

5.2.3 AI 対応のテスト組織におけるテストプロセスの進化

生成 AI の統合により、テスト組織におけるテスト担当者およびテストマネージャーの従来のテストプロセスは変革される。

テスト担当者は、テスト設計とテスト実行のスペシャリストから AI 支援テストのスペシャリストに進化し、テスト技術の専門知識と、AI が生成したテストウェアをガイドおよび検証するスキルを組み合わせる。彼らのテストタスクは、AI ベースの出力全体のレビュー、プロンプトの改良、テスト固有のプロンプトライブラリのメンテナンスを含むように拡張される。

テストマネージャーの責任は、AI ベースのテスト戦略の開発、AI ベースのリスク管理、および AI ベースのテストプロセスの監視と制御を含むように拡張される。テストマネージャーは、人間と AI の能力のバランスを取ること、ユースケースに応じた AI ガバナンスの枠組みを確立すること、テストチームが従来のテスト能力と AI リテラシーの両方を維持できるようにすることに重点を置いている。テストマネージャーは人間のテスト担当者を率いるだけでなく、生成 AI を利用したテストエージェントとの調整も行うため、人々と生成 AI ツールのハイブリッドチームを監督するための新しいマネジメントスキルが必要になる。

6 参考文献

標準

- **ISO/IEC 42001:2023** (2023)、Information technology - Artificial intelligence- Management system
- **ISO/IEC 23053:2022** (2022)、Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)

ISTQB® ドキュメント

- **[ISTQB_CTFL_SYL]** ISTQB® ファウンデーションレベルシラバス v4.0、2023 年

用語集の参照

- **ISTQB® 用語集** <https://glossary.istqb.org/>

本

- Winteringham M. (2024) Software Testing with Generative AI、Manning Publications (2025 年 3 月 5 日)、ISBN-13: 978-1633437364、2024 年 12 月 10 日 - 304 ページ

記事

- (Berthelot 2024) Berthelot, Adrien、他。「LCA ベースの方法論を使用した Generative-AI サービスの環境への影響の推定」 *プロセディア CIRP* 122 (2024) : 707-712
- (Gallegos 2024) Gallegos, Isabel O.、他。「大規模言語モデルにおけるバイアスと公平性: 調査」 *計算言語学* (2024) : 1-79
- (Li 2024) Yihao Li, Pan Liu, Haiyang Wang, Jie Chu, W. Eric Wong、ソフトウェアテストのための大規模言語モデルの評価、コンピューター標準とインターフェース (2024)、doi:<https://doi.org/10.1016/j.csi.2024.103942>
- (Luccioni 2024a) Luccioni, Sasha, Yacine Jernite, Emma Strubell。「電力を大量に消費する処理: AI 導入コストを押し上げるワット数?」 公平性・説明責任・透明性に関する 2024 年 ACM 会議、2024 年
- (Mailach 2024) Mailach 「本番用の LLM 駆動型アプリケーションの構築に関する実務者によるディスカッション」 *arXiv プレプリント arXiv:2411.08574* (2024)
- (Mirzadeh 2024) Mirzadeh。「GSM-Symbolic: 大規模言語モデルにおける数学的推論の限界を理解する」 *ArXiv abs/2410.05229* (2024)
- (NIST AIRMF 1.0) 米国標準技術研究所。人工知能リスク管理フレームワーク (AIRMF 1.0)。NIST AI100-1、米国商務省、2023 年、<https://doi.org/10.6028/NIST.AI.100-1>
- (Parthasarathy 2024) Parthasarathy, Venkatesh Balavadhani、他。「基本からブレイクスルーまで LLMs をファインチューニングするための究極のガイド: テクノロジー、研究、ベストプラクティス、応用研究の課題と機会を徹底的にレビューする。」 *arXiv プレプリント arXiv:2408.13296* (2024)

- (Schulhoff, 2024) Schulhoff, 「プロンプトレポート: プロンプト技法の系統的調査」、Art no. arXiv:2406.06608、2024。doi:10.48550/arXiv.2406.06608
- (Shuyin 2023) Ouyang, Shuyin, 他。 「LLM はチョコレートの箱に似ている: コード生成における ChatGPT の非決定性」 arXiv プレプリント arXiv:2308.02828 (2023)
- (M Sinha 2024) M Sinha。 「LLMOps: 定義、フレームワーク、ベストプラクティス」 2024 年電気、コンピューター、エネルギー技術に関する国際会議 (ICECET) . IEEE、2024.
- (Wang 2024) Wang, Yanlin, 他。 「ソフトウェアエンジニアリングのエージェント: 調査、ランドスケープ、ビジョン」。 arXiv プレプリント arXiv:2409.09030 (2024)
- (文漢 2024) 王文漢ほか。 「TESTEVAL: テストケース生成のための大規模言語モデルのベンチマーク」。 arXiv プレプリント arXiv:2406.04531 (2024)
- (Zhao 2024) Zhao, Penghao, 他。 「AI 生成コンテンツの検索拡張生成: 調査」 arXiv プレプリント arXiv:2402.19473 (2024)

ウェブページ

(AI 法 2024) 欧州委員会。 「人工知能に対するヨーロッパのアプローチ」 欧州のデジタル未来の形成、欧州委員会、<https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence> 2024 年 11 月 24 日にアクセス。

Heikkilä 2023) Heikkilä, M. (2023 年 12 月 1 日)。生成 AI を使用して画像を作成すると、携帯電話を充電するのと同じくらい多くのエネルギーが消費される。 MIT テクノロジーレビュー。 から取得。
<https://www.technologyreview.com/2023/12/01/1084189/making-an-image-with-generative-ai-uses-as-much-energy-as-charging-your-phone/>

(ルッチョーニ 2024b) ルッチョーニ, S. (2024 年 2 月 22 日)。生成 AI の環境コストは高騰している。自然。 から取得。
<https://www.nature.com/articles/d41586-024-00478-x>

(Google 開発用語集 2024) Google 開発者。 (未確認)。機械学習用語集: 生成 AI。 2024 年 11 月 24 日に取得、以下か
<https://developers.google.com/machine-learning/glossary/generative> ら

(MIT 2024) 「用語集: 生成 AI の基礎」。 *MIT Sloan Teaching & Learning Technologies*、MIT Sloan School of Management、<https://mitsloanedtech.mit.edu/ai/basics/glossary> 2024 年 11 月 24 日にアクセス。

前述の参考文献は、インターネットなどで入手可能な情報を示している。このシラバスの発行時にこれらの参考文献が確認されている場合でも、その参考文献が入手できなくなった場合、ISTQB® は責任を負わない。

7 付録 A - 学習の目的/知識の認知レベル

このシラバスに適用される具体的な学習の目的は、各章の冒頭に示されている。シラバスの各トピックは、その学習目的に応じて検討される。

学習の目的は、以下にリストされているように、知識の認知レベルに対応する動作動詞から始まる。

レベル 1: 記憶 (K1)

受験者は用語や概念を記憶し、認識し、想起することができる。

動作動詞: 想起する、認識する。

例
テストピラミッドの概念を想起する
テストの典型的な目的を認識する

レベル 2: 理解 (K2)

受験者は、トピックに関連するステートメントの理由や説明を選択し、テストの概念を要約、比較、分類し、例を示すことができる。

動作動詞: 分類する、比較する、区別する、区別する、説明する、例を挙げる、解釈する、要約する

例	注意事項
テストツールをその目的と支援するテストアクティビティに応じて分類する	
さまざまなテストレベルを比較する	類似点、相違点、またはその両方を探すために使用できる
テストとデバッグを区別する	概念間の違いを探す
プロジェクトのリスクと製品のリスクを区別する	2つ（またはそれ以上）の概念を個別に分類できる
コンテキストがテストプロセスに及ぼす影響を説明する	
なぜテストが必要なのかを例を挙げて説明する	
特定の障害プロファイルから欠陥の根本原因を推測する	
作業成果物のレビュープロセスの活動を要約する。	

レベル 3: 適用 (K3)

受験者は、馴染みのあるタスクに直面したときに手順を実行することも、正しい手順を選択して特定の状況に適用することもできる。

アクション動詞: 適用、実装、準備、使用

例	注意事項
境界値分析を適用して、指定された要件からテストケースを導き出す	手順/技法/プロセスなどを参照する必要がある
メトリクス収集方法を実装して、技術要件と管理要件を支援する	
モバイルアプリのインストール可能性テストを準備する	
トレーサビリティを使用して、テストの進行状況を監視し、テストの目的、テスト戦略、テスト計画との完全性および一貫性を確認する	候補者に技術や手順を使用できるようにすることを求める LO で使用できる。「適用」に似ている

参照

(学習の目的の認知レベルの場合)

Anderson, L.W. および Krathwohl, D.R (編) (2001) 学習、教育、および学習のための分類法

評価：ブルームの教育目標分類法の改訂、アリン&ベーコン

8 付録 B - 学習の目的を含むビジネス成果トレーサビリティマトリックス

このセクションでは、生成 AI を使用した認定テスターテストのビジネス成果と学習の目的の間のトレーサビリティを列挙する。各 HO は単一の LO に関連付けられているため、この表ではハンズオン目標については言及されていない。HO と BO 間のトレーサビリティは、HO が関連付けられている LO を介して行われる。

ビジネスの成果: 生成 AI を使用した認定テスターのテスト			BO1	BO2	BO3	BO4	BO5
GenAI-BO1	生成 AI の基本的な概念、機能、制限を理解する		8				
GenAI-BO2	ソフトウェアテスト用に大規模な言語モデルをプロンプトする実践的なスキルを開発する			10			
GenAI-BO3	ソフトウェアテストに生成 AI を使用するリスクと軽減策についての洞察を得る				11		
GenAI-BO4	ソフトウェアテスト用の生成 AI ソリューションのアプリケーションについての洞察を得る					19	
GenAI-BO5	組織内でのソフトウェアテストの生成 AI 戦略とロードマップの定義と実装に効果的に貢献する						13

ビジネスの成果: 生成 AI を使用した認定テストのテスト			BO1	BO2	BO3	BO4	BO5
ユニーク LO	学習の目的	K レ ベル					
1	ソフトウェアテストのための生成 AI の概要						
1.1	生成 AI の基盤と主要な概念						
GenAI-1.1.1	さまざまなタイプの AI を想起する: シンボリック AI、古典的な機械学習、ディープラーニング、生成 AI	K1	X				
GenAI-1.1.2	生成 AI と大規模言語モデルの基本を説明する	K2	X				
GenAI-1.1.3	基盤大規模言語モデル（ファウンデーションモデル）、インストラクションチューニングモデルと推論 LLM を区別する	K2	X				
GenAI-1.1.4	マルチモーダル LLM と視覚言語モデルの基本原則を要約する	K2	X				
1.2	ソフトウェアテストにおける生成 AI の活用: 基本原則						
GenAI-1.2.1	テストタスクの主要な LLM 能力の例を示す	K2	X			X	
GenAI-1.2.2	ソフトウェアテストに生成 AI を使用する場合の対話モデルを比較する	K2	X			X	

ビジネスの成果: 生成 AI を使用した認定テストのテスト			BO1	BO2	BO3	BO4	BO5
2	効果的なソフトウェアテストのためのプロンプトエンジニアリング						
2.1	効果的なプロンプト開発						
GenAI-2.1.1	ソフトウェアテスト用の生成 AI で使用されるプロンプトの構造の例を挙げる	K2		X			
GenAI-2.1.2	ソフトウェアテストのためのコアプロンプト技法を区別する	K2		X			
GenAI-2.1.3	システムプロンプトとユーザープロンプトを区別する	K2		X			
2.2	ソフトウェアテストタスクへのプロンプトエンジニアリング技法の適用						
GenAI-2.2.1	生成 AI をテスト分析タスクに適用する	K3		X			
GenAI-2.2.2	生成 AI をテスト設計およびテスト実装タスクに適用する	K3		X			
GenAI-2.2.3	自動リグレッションテストに生成 AI を適用する	K3		X			
GenAI-2.2.4	生成 AI をテスト制御および監視タスクに適用する	K3		X			

ビジネスの成果: 生成 AI を使用した認定テスターのテスト			BO1	BO2	BO3	BO4	BO5
GenAI-2.2.5	特定のテストタスクに対して、状況に応じたプロンプト技法を選択して適用する	K3		X		X	
2.3	生成 AI の結果を評価し、ソフトウェアテストタスクのプロンプトを調整する						
GenAI-2.3.1	テストタスクにおける生成 AI の結果を評価するためのメトリクスを理解する	K2		X	X	X	
GenAI-2.3.2	プロンプトを評価し、反復的に調整するための技法の例を示す	K2		X	X	X	
3	ソフトウェアテストにおける生成 AI のリスク管理						
3.1	ハルシネーション、推論エラー、バイアス						
GenAI-3.1.1	生成 AI システムにおけるハルシネーション、推論エラー、バイアスの定義を想起する	K1	X		X	X	
GenAI-3.1.2	LLM 出力のハルシネーション、推論エラー、バイアスを分析する	K3			X	X	
GenAI-3.1.3	ソフトウェアテストタスクにおける生成 AI ハルシネーション、推論エラー、バイアスを軽減する技法をまとめる。	K2			X	X	
GenAI-3.1.4	LLM の非決定的動作に対するリコール軽減技法	K1	X		X	X	

ビジネスの成果: 生成 AI を使用した認定テストのテスト			BO1	BO2	BO3	BO4	BO5
3.2	ソフトウェアテストにおける生成 AI のデータプライバシーとセキュリティリスク						
GenAI-3.2.1	ソフトウェアテストでの生成 AI の使用に関連する主要なデータプライバシーとセキュリティリスクについて説明する	K2			X	X	
GenAI-3.2.2	ソフトウェアテストで生成 AI を使用する場合のデータプライバシーと脆弱性の例を挙げる	K2			X	X	
GenAI-3.2.3	データプライバシーを保護し、ソフトウェアテスト用の生成 AI のセキュリティを強化するための緩和戦略を要約する。	K2			X	X	
3.3	ソフトウェアテストにおける生成 AI のエネルギー消費と環境への影響						
GenAI-3.3.1	ソフトウェアテストにおける生成 AI のエネルギー消費に対するタスクの特性とモデルの使用の影響を説明する	K2			X	X	
3.4	AI の規制、標準、ベストプラクティスフレームワーク						
GenAI-3.4.1	ソフトウェアテストにおける生成 AI に関連する AI 規制、標準、ベストプラクティスフレームワークの例を想起すること	K1			X	X	X
4	LLM 搭載ソフトウェアテスト用テストインフラストラクチャ						
4.1	LLM を利用したテストインフラストラクチャのアーキテクチャアプローチ						

ビジネスの成果: 生成 AI を使用した認定テストターのテスト			BO1	BO2	BO3	BO4	BO5
GenAI-4.1.1	LLM を利用したテストインフラストラクチャの主要なアーキテクチャコンポーネントと概念について説明する	K2				X	X
GenAI-4.1.2	RAG を要約する	K2				X	X
GenAI-4.1.3	テストプロセスの自動化における LLM 搭載エージェントの役割と応用について説明する	K2				X	X
4.2	ファインチューニングと LLMops: ソフトウェアテスト用の生成 AI の運用化						
GenAI-4.2.1	特定のテストタスクに対する言語モデルのファインチューニングについて説明する	K2				X	X
GenAI-4.2.2	LLMops と、テストタスク用の LLM の展開と管理におけるその役割について説明する	K2				X	X
5	テスト組織への生成 AI の導入と統合						
5.1	ソフトウェアテストにおける生成 AI の導入のロードマップ						
GenAI-5.1.1	シャドーAI のリスクを想起する	K1					X
GenAI-5.1.2	ソフトウェアテストの生成 AI 戦略を定義する際に考慮すべき重要な側面について説明する	K2					X

ビジネスの成果: 生成 AI を使用した認定テストターのテスト			BO1	BO2	BO3	BO4	BO5
GenAI-5.1.3	特定のコンテキストでソフトウェアテストタスクに LLM/SLM を選択するための主要な基準を要約する	K2					X
GenAI-5.1.4	テスト組織での生成 AI の導入における主要な段階を想起すること	K1					X
5.2	ソフトウェアテストに生成 AI を採用する際の変更の管理						
GenAI-5.2.1	テストプロセスで生成 AI を効果的に使用するためにテスト担当者に必要な必須のスキルと知識領域を説明する	K2					X
GenAI-5.2.2	テスト活動における生成 AI の導入を支援するために、テストチーム内で AI スキルを育成するための戦略を想起する	K1					X
GenAI-5.2.3	生成 AI を導入する際に、テスト組織内でテストのプロセスと責任がどのように変化するかを認識する	K1					X

9 付録 C - リリースノート

今回のバージョンは V1.0 である。この最初のバージョンにはリリースノートはない。

10 付録 D - 生成 AI 固有の用語

用語名	意味
AI チャットボット	LLM を使用してクエリを処理し、人間のようなテキスト応答を生成することで、ユーザーとの会話的なコミュニケーションを可能にする対話型エージェント。
コンテキストウィンドウ	言語モデルが応答を生成するときに考慮する、トークンで測定されるテキストの範囲。出力の関連性と一貫性に影響する。
ディープラーニング	複数の層を持つニューラルネットワークを使用する ML。
エンベディング	トークンを連続空間内の密なベクトルとして表現するために使用される技法。意味論的、構文的、および文脈的な関係を捉えるようにトレーニング中に学習される。
特徴量	ML アルゴリズムによるトレーニングと ML モデルによる予測に使用される入力データの個々の測定可能な属性
Few-Shot プロンプト	プロンプト内でモデルにいくつかの例を与え、モデルが適切な応答を生成できるようにする技法。
ファインチューニング	ラベル付けされた事例のデータセットを使用して LLM の重みを更新し、特定のタスクまたはドメインに適応させる教師あり学習プロセス。
基盤大規模言語モデル	幅広いテキストデータで事前学習された汎用モデルで、学習した言語パターンに基づいて次の単語を予測できる。 別名:基盤 LLM (訳注: 原文では、BaseLLM または Foundation LLM と記述されている)
生成 AI (GenAI)	機械学習モデルを用いて、人間が作成したコンテンツに似た (新しい) 知的コンテンツを生成する人工知能システムの種類。
生成的事前学習変換器 (GPT)	人間のようなテキストを理解して生成するために、膨大な量のテキストデータで事前トレーニングされた、トランスフォーマーベースの深層学習モデルの種類。
ハルシネーション	LLM によって作成された誤った情報。
インストラクションチューニング LLM	指示に従うように訓練された基盤 LLM。多くの場合は正解を促すフィードバックによって強化される。
大規模言語モデル (LLM)	膨大な量の言語データを用いて、人間と同じような方法でテキストを理解して生成するコンピュータープログラム。
LLM 搭載エージェント	LLM 推論、意思決定、記憶を統合し、ツールを用いてタスクを実行するアプリケーション。

LLMOps	本番環境で LLM を導入、監視、保守することに重点を置いた実践方法とツール。
機械学習 (ML)	計算技術を用いて、システムがデータまたは経験から学習できるようにするプロセス (ISO/IEC TR 29119-11)。
メタプロンプト	機能を探索または自動化するための特定のプロンプトを生成する、より高いレベルの命令の作成。
マルチモーダルモデル	テキスト、画像、音声など、複数のデータタイプにわたるコンテンツを処理および生成できる生成 AI モデル。
自然言語処理 (NLP)	情報を取得したり知識を表現したりするために、コンピューターによって自然言語でエンコードされたデータを処理すること。
ワンショットプロンプト	LLM の応答を導くために例を 1 つプロンプトに含めるプロンプト作成技法。
プロンプト	生成 AI および大規模言語モデルで特定の応答を引き出すために提供される自然言語入力。
プロンプトチェーン	あるプロンプトの出力を別のプロンプトの入力として使用し、一連のプロンプトを作成するプロンプト技法。
プロンプトエンジニアリング	LLM が望ましい出力を生成できるように入力プロンプトを設計および改良するプロセス。
推論 LLM	人間のような推論プロセスを模倣する能力を洗練させることで、指導に合わせて調整されたモデルを基盤とする LLM。
検索拡張生成 (RAG)	LLM の機能とリトリバーを組み合わせる関連データを取得し、正確で文脈に応じた応答を生成する技法。
シャドー AI	正式な承認や管理を受けずに、組織内で生成 AI ツールまたはシステムを使用すること。
小規模言語モデル (SLM)	意図的に小型化するように設計およびトレーニングされた言語モデル。効率とタスク固有の言語理解のバランスを提供する。
シンボリック AI	記号、ルール、構造化された知識を使用して推論をモデル化する AI アプローチ。
システムプロンプト	事前定義された命令セット。通常はチャットボットのユーザーには非表示になっており、LLM の応答の文脈、トーン、境界を一貫して確立し、対話全体を通じてその動作を手引きする。
Temperature	LLM の出力のランダム性または創造性を制御するパラメータ。
トークン化	言語モデルによる処理のために、テキストをより小さな単位に分割するプロセス。
トランスフォーマー	自己注意メカニズムを利用して入力シーケンス内の長距離依存関係を捕捉する深層学習モデルアーキテクチャ。

ユーザープロンプト	ユーザーが大規模言語モデル (LLM) に入力する命令またはクエリ。特定のタスクを実行するか、必要な情報を提供するようにモデルの応答を指示する。
ベクトルデータベース	データの高次元ベクトル表現を保存およびクエリするために最適化されたデータベース。
視覚言語モデル	視覚データとテキストデータを共同処理して、両方の形式にわたってコンテンツをリンクおよび生成することでタスクを実行する生成 AI システム。
ゼロショットプロンプト	プロンプトに例が含まれておらず、モデルの既存の知識に依存して応答を生成する、プロンプト作成技法。

11 付録 E - 商標

ISTQB® は International Software Testing Qualifications Board の登録商標である